Technical Report - **Product specification**

STS – Smart Training System

Course:	IES – Introduction to Software Engineering
Date:	Aveiro, 07/12/2024
Students:	112981: Tomás Santos Fernandes 113384: Danilo Micael Gregório Silva 115304: Pedro Miguel Azevedo Pinto 104384: João Pedro Azevedo Pinto
Project abstract:	**Smart Training System** is a web-based platform that will be designed to assist coaches in monitoring and analyzing player performance in real-time. The application will provide key physical statistics, including heart rate, body temperature and respiratory rate. Utilizing distributed data streams from sensors, the platform will enhance real-time decision-making during training and matches. Key features include real-time data integration, role-based access control and a REST API for the web portal and future mobile applications.

Table of contents:

1 Introduction

STS – Smart Training System is a digital solution intended to help coaches, team directors, personal trainers, and players to oversee and assess player performance as it happens. This project is being developed as part of the Introduction to Software Engineering (IES) course. The main objective of this assignment is not solely the delivery of a final product, but rather the focus on applying software development processes and methodologies. To effectively manage our development process, we will implement tools and practices such as **backlog management** for tracking tasks, **Docusaurus** for extra and simple documentation and **GitHub Workflow** defined for each commit, issue, branches and pull requests. **Weekly meetings** will be held to facilitate organization and communication within the team and can be located in Docusaurus' documentation.

By prioritizing collaboration and clear communication, we aim to maintain a structured workflow that emphasizes continuous testing, integration, and validation of each module while ensuring alignment with the project objectives.

2 Product concept

Vision statement

Smart Training System is a web-based application intended to support **coaches**, **team directors**, **personal trainers**, and **players** in real-time monitoring and analysis of player performance. It addresses the need for immediate access to crucial metrics such as **heart rate, body temperature and respiratory rate**. By providing these insights, the platform enhances decision-making during training and matches, allowing coaches to optimize player performance and strategies effectively.

The **primary problem** our system aims to solve is the lack of accessible real-time performance data, which limits coaches' and trainers' ability to make informed adjustments. With features that include **user access management** for team directors, **performance statistics** for coaches, and **historical tracking** for players, the STS platform focuses on delivering targeted functionality that meets the specific needs of each user role.

The project **initially** aimed to offer extensive features for football fans, including comprehensive coverage of the English Football League and the Premier League. However, the **focus has now shifted** towards addressing the needs of coaches, team directors, and personal trainers from **clubs worldwide** who seek **advanced analysis tools to improve decision-making** during training sessions and matches.

The STS platform will **differentiate** itself by emphasizing **real-time data** collection and user-friendly interfaces tailored for coaches, team directors, personal trainers, and players. This focus enables immediate feedback and adjustments, setting it apart from **existing solutions** that primarily **rely on post-game** and in-train analysis.

The **project concept** emerged from team collaboration and insights from professors, ensuring relevance to practical needs. User stories outlining features like permission management, real-time performance tracking, and injury alerts directed the development of a system that effectively meets user requirements.

Personas and Scenarios

Personas

The design and development of the **Smart Training System** were supported by several Personas that represent the key users of the application: a primary Persona, **Carlos**, a Football Team Coach (Box 1); a secondary Persona, **Ricardo**, a Personal Trainer (Box 2); and one served Persona, **Miguel** (Box 3) a football player. Additionally, an administrative Persona, **João, the Team Director** (Box 4), is included to represent the managerial aspects of the system, and **David, the System Administrator** (Box 5), to represent the technical and security management of the application.

Figure 1. Primary Persona: **Carlos**, the Football Team Coach.

Carlos is a 45-year-old football coach with over 20 years of experience in coaching youth and professional teams. He is passionate about football tactics and player development. Outside of football, Carlos enjoys reading historical novels and spending time with his family.

Carlos is **tech-savvy** and always looking for new ways to gain a competitive edge. He believes that **real-time data and analytics** can significantly enhance team performance. However, he finds it challenging to monitor all players' physical statistics during training and matches due to the lack of integrated tools.

MOTIVATION: Carlos wants to utilize a system that provides real-time access to his players' physical and performance data, allowing him to make quick decisions during training and matches to optimize team performance.

Figure 2. Secondary Persona: **Ricardo**, the Personal Trainer.

Ricardo is a 30-year-old personal trainer who has been working with the team for the past five years. He is energetic and highly dedicated to improving each player's physical fitness. In his free time, **Ricardo** enjoys hiking and practicing yoga.

He uses various fitness tracking devices but finds it cumbersome to collect and analyze data from multiple sources. **Ricardo** aims to personalize training sessions based on each player's specific needs but lacks an efficient way to **monitor and adjust these sessions in real time.**

MOTIVATION: Ricardo seeks a solution that allows him to monitor players' physical statistics during training, receive alerts for any critical changes, and adjust training programs accordingly.

Figure 3. Served Persona: **Miguel**, a Professional Football Player.

Miguel is a 25-year-old forward who has been playing professionally for seven years. He is ambitious and constantly strives to improve his skills. Miguel enjoys **analyzing his performance stats** to identify areas for improvement.

However, Miguel often feels limited by the **lack of detailed performance data**, believing that access to comprehensive insights could significantly enhance his career development.

MOTIVATION: Miguel wants to access his physical and performance statistics to identify his strengths and weaknesses and set new personal goals.

Figure 4. Administrative Persona: **João**, the Team Director.

João is a 50-year-old team director responsible for the administrative and managerial functions of the club. He has a background in sports management and is keen on implementing systems that **enhance operational efficiency.** João enjoys playing golf and attending networking events.

He is responsible for managing user access levels within the club's systems. João wants to ensure that sensitive information is only accessible to authorized personnel and that new players are onboarded smoothly.

MOTIVATION: João seeks an efficient way to manage user permissions, add or remove team members, and ensure that everyone has appropriate access to the resources they need.

Figure 5. Administrative Persona: **David**, the System Administrator.

David is a 40-year-old system administrator who has been managing IT infrastructure for sports organizations for the past 10 years. He is highly skilled in system security, network management, and ensuring the smooth operation of **software** platforms. In his free time, David enjoys photography and participates in online security forums to stay updated on the latest trends in cybersecurity.

David oversees the backend of the Smart Training System, managing tasks such as integrating new sensors, monitoring system performance, and ensuring security and operational standards are met. He routinely reviews activity logs and tracks the total number of requests to various endpoints to ensure the system operates efficiently and reliably.

MOTIVATION: David is dedicated to ensuring the Smart Training System operates efficiently while optimizing its settings to adapt to the club's evolving requirements.

Scenarios

The following context scenarios illustrate how the **Smart Training System** integrates into the daily activities of its users, helping them achieve their goals.

Scenario 1: Carlos Utilizes Real-Time Player Data

Carlos is preparing for an important match. During the training session, he opens the Smart Training System on his tablet. He navigates to the real-time dashboard displaying each player's heart rate, body temperature and respiratory rate. Noticing that Miguel's heart rate is unusually high, Carlos decides to substitute him to prevent injury and maintain training intensity.

Underlined Actions: opens the Smart Training System; navigates to the real-time dashboard; analyzes specific player statistics; decides to substitute him.

Scenario 2: Ricardo Personalizes Training Sessions

Ricardo plans a specialized training session for a group of players recovering from minor injuries. He logs into the system and selects the players involved. During the session, he receives an alert that Rui has exceeded his safe heart rate zone. Ricardo promptly adjusts the exercise intensity.

Underlined Actions: opens the Smart Training System; selects the players; receives an alert; adjusts the exercise intensity.

Scenario 3: Miguel Reviews Performance and Sets Goals

After a training session, Miguel accesses the Smart Training System on his smartphone. He reviews his performance statistics.

Underlined Actions: opens the Smart Training System; reviews performance statistics.

Scenario 4: João Manages Team Access

João receives notification that a new player has joined the team. He logs into the Smart Training System, navigates to the team management panel, and adds the new player to the roster. João assigns the appropriate access level, ensuring the player can view only their personal data.

Underlined Actions: opens the Smart Training System; goes to the team management panel; adds the new player; assigns access levels.

Scenario 5: Carlos Analyzes Past Match Data

Before an upcoming match against a strong opponent, Carlos wants to revisit past games. He accesses previous match statistics, focusing on players' physical and performance data. Carlos identifies patterns that could be improved and develops a new strategy. He keeps these insights confidential within the system.

Underlined Actions: opens the Smart Training System; accesses previous match statistics; identifies patterns; develops a new strategy; keeps insights confidential.

Scenario 6: David Assigns Permissions to a Team Director

David receives notification that a new team director, João, has been hired and needs access to the Smart Training System. He logs onto the administrative dashboard and navigates to the team management section. David creates a new user account for João and assigns him the role of Team Director, carefully configuring the permissions to allow João to autonomously manage the access levels of his team members without accessing sensitive system settings.

Underlined Actions: logs into the administrative dashboard; navigates to the team management section; creates a new user account; assigns the Team Director role; configures permissions.

Scenario 7: David Checks System Status

As part of his daily routine, David logs into the Smart Training System's admin dashboard to ensure everything is running smoothly. He checks the system status and sees that all services are operational without any issues. After verifying that there are no alerts or warnings, David logs out, confident that the system is performing as expected.

Underlined Actions: logs into the admin dashboard; checks the system status; verifies no alerts; logs out.

Product requirements (User stories)

Building on the defined Personas and Scenarios, a set of requirements was established to guide the development of the Smart Training System (see below). These requirements focused on providing real-time access to players' physical and performance data for coaches and enabling team directors to manage user access levels autonomously.

1. **As an administrator**, I want to be able to **assign permissions to a team director** so that he can autonomously manage the access levels of his team members.
2. **As an administrator**, I want to **monitor the system's performance,** including the number of requests made to each endpoint, to ensure the application's efficient and secure operation.
3. **As an administrator**, I want to **monitor the system's performance,** including the number of sensors data, to ensure the application's efficient and secure operation.
4. **As an administrator**, I want to manage teams by viewing all existing teams, creating new teams, and seeing the team directors associated with each team, to maintain organizational

oversight and ensure efficient team management.

5. **As an administrator**, I want to manage sensors by adding new sensors with a specified *sensorId* and viewing the players assigned to each sensor, as well as removing sensors from the app, to ensure the system stays up to date.
6. **As a team director**, I want to **manage each user's access levels in the application** to ensure that each member has access only to information relevant to their role.
7. **As a team director**, I want to **add new players to the team to facilitate the work** of the coach, personal trainer, and the players themselves in managing information and training.
8. **As a team director**, I want to **remove access for members who are no longer part of the team**, such as in cases of dismissals or transfers, to maintain the security and integrity of the team's information.
9. **As a team director,** I want to access a list of sensors assigned to my team, so that I can assign or remove sensors from players.
10. **As a coach**, I want to view **real-time physical and performance statistics of my team players**, including heart rate, body temperature and respiratory rate, to make quick decisions during training and optimize player performance.
11. **As a coach,** I want to receive alerts when a player exceeds certain physical limits (e.g., maximum heart rate), to intervene immediately and prevent injuries.
12. **As a coach,** I want to initiate a session with selected players with selected sensors from my team, to personalize activities and focus on each player's specific goals and metrics.
13. **As a coach,** I want to have exclusive authority to start a match session, where I can specify which players will be tracked by the sensors.
14. **As a coach**, I want to access a list of sensors assigned to my team, so that I can assign or remove sensors from players.
15. **As a coach**, I want to **compare the statistics of two or more players on my team** to understand how their performance compares and make the best strategic decisions for the team.
16. **As a coach**, I want to view detailed session data and player performance, so that I can track each player's progress and identify areas for improvement.
17. **As a personal trainer,** I want to receive alerts when a player exceeds certain physical limits (e.g., maximum heart rate), to intervene immediately and prevent injuries.
18. **As a personal trainer**, I want to **consult all the physical and performance statistics of my team players during training** to monitor progress and adjust sessions as needed.
19. **As a personal trainer**, I want to **initiate training with selected players** with **selected sensors** from my team to personalize activities and focus on each player's specific goals and metrics.
20. **As a personal trainer**, I want to access a list of sensors assigned to my team, so that I can assign or remove sensors from players.
21. **As a player,** I want to **view the history of my performance throughout the season** to track my progress and set new goals.
22. **As a player,** I want to view the sessions I participated in and see detailed performance metrics, so that I can track my progress and understand my strengths and areas for improvement.

3 Architecture notebook

Key requirements and constraints

There are several key requirements and constraints that significantly impact the design and architecture of the system. These include performance needs, integration with external systems, scalability, and long-term maintainability. The architecture must address these critical factors to ensure smooth operation and adaptability.

Real-Time Data Ingestion and Processing:

- The system will handle **real-time data streams** from sensors. This requires the architecture to support **high-throughput message processing** with minimal latency. To address this, **Kafka** is used to manage and distribute tasks asynchronously, allowing the backend to process large volumes of data without bottlenecks.
- **Key Constraint**: The architecture must handle spikes in data streams and manage processing effectively under conditions where large amounts of sensor data are ingested continuously.

User Interfaces:

- The chosen frontend technologies (**React**, **Tailwind CSS**) ensure responsive and cross-platform compatibility, making the system accessible from different user devices.
- **Key Requirement**: The system must maintain a consistent and optimized user experience across different platforms (desktop, mobile, large screen displays).

Data Storage and Scalability:

- The system will store both relational data (e.g., user profiles) and **time-series data** (e.g., sensor readings). **PostgreSQL** is used for relational data, and **TimescaleDB** is integrated for efficient time-series data handling. The architecture needs to ensure **scalability**, allowing for the storage and querying of large datasets as the number of users and data points grows.
- **Key Constraint**: The system must efficiently store and retrieve data, especially for time-based queries, without compromising performance as the data grows over time.

High Availability:

- The system must be robust enough to handle increased loads, especially during peak times, such as during real-time data monitoring. Nginx serves as a **reverse proxy, caching layer, and endpoint monitoring tool**.
- **Key Requirement**: The architecture must support **horizontal scalability** to handle high-traffic periods without downtime.

Modular and Maintainable Architecture:

- The architecture must be designed for **long-term maintenance** and future expansion. With the use of **Spring Boot** and **Docker**, the system can be containerized, making it easier to deploy updates, isolate components, and manage dependencies. This also ensures that individual services (e.g., Kafka, databases) can be scaled or replaced without disrupting the entire system.
- **Key Requirement**: The architecture must be modular to allow for the independent deployment of services, minimizing downtime and allowing for the evolution of the system

over time.

Performance Monitoring and Logging:

- As the system will run in a production environment, it is essential to **monitor performance** and **logs** in real-time. The **EK Stack** (Elasticsearch, Kibana) must be integrated to provide **real-time monitoring**, **log aggregation**, and **troubleshooting**. This enables the development team to track performance metrics and quickly resolve any issues that arise in production.
- **Key Requirement**: The system must provide robust **monitoring and logging** capabilities, enabling the team to keep track of system health, detect anomalies, and address performance bottlenecks.

Dockerized Deployment:

- The system needs to be deployable across multiple environments (development, staging, production) with minimal configuration changes. The use of **Docker** ensures that each component (backend, frontend, databases) runs in isolated containers, making it easier to maintain consistency across environments and simplifying the deployment process.
- **Key Requirement**: The system must be easily deployable and portable across environments, ensuring consistent behavior and performance from development to production.

Architectural view

Below is a high-level diagram illustrating the system architecture of the **Smart Training System**, showing how the various components and technologies are integrated to form a cohesive and scalable platform.

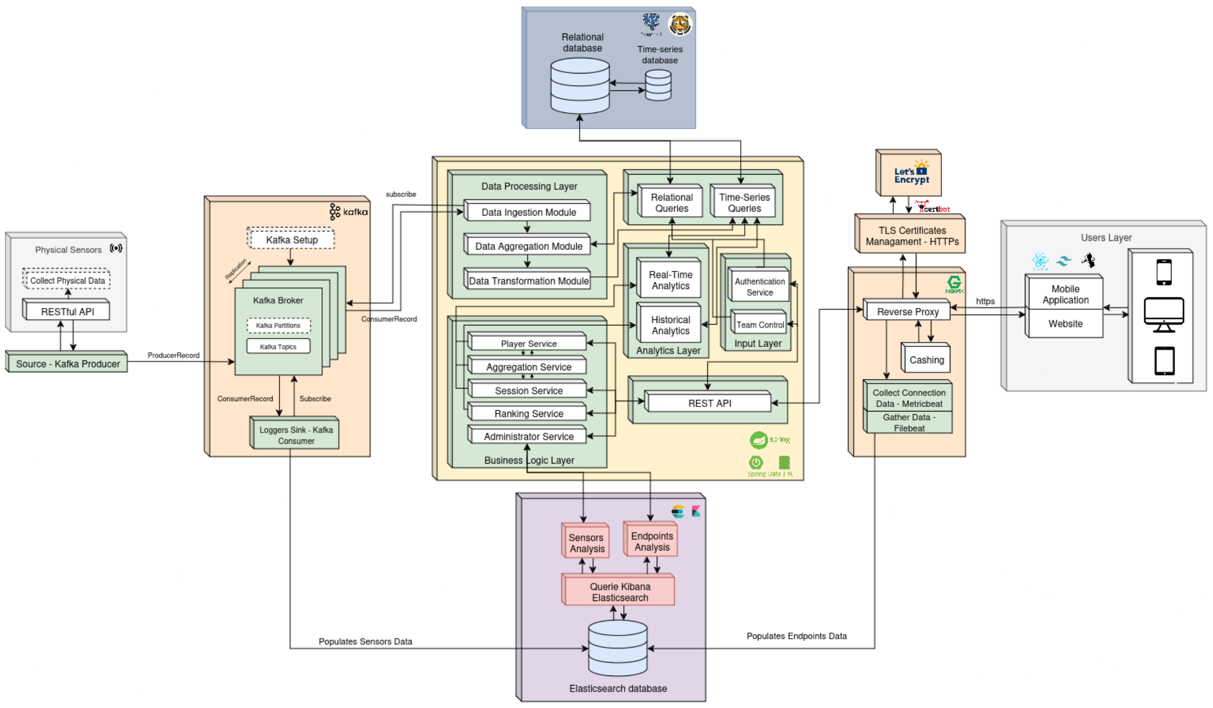


Figure 6. Project Architecture Design.

This architecture represents a **modern, scalable web application** using a **decoupled services** approach and **Data-driven design**. The architecture is divided into multiple layers to ensure modularity, scalability, and maintainability:

- **Frontend**: The user interface is built using **ReactJS**, a component-based JavaScript framework, enhanced with **Tailwind CSS** for responsive and customizable styling. **Vite** serves as the development build tool, providing fast reloads and optimized production builds. The frontend communicates with the backend via **RESTful APIs**.
- **Backend**: The backend is developed using **Spring Boot**, which handles API requests through **Spring Web (REST API)** and manages the business logic in the **Service Layer**. The backend processes incoming data, connects to databases, and integrates with **Kafka** for asynchronous message handling.
- **Data Layer**: Data is stored and managed in two main databases: **PostgreSQL** for relational data (e.g., user information) and **Timescale DB** (integrated in PostgreSQL) for efficiently handling time-series data (e.g., sensor readings). **Spring Data JPA** is used for seamless database integration and query handling.
- **Message Queuing**: **Kafka** serves as a message broker, handling asynchronous tasks such as processing sensor data. It decouples the data sources from the backend, ensuring that high volumes of incoming data can be processed efficiently without overloading the system.
- **Nginx**: As the **reverse proxy**, **Nginx** handles all incoming requests, serving static files (like the React frontend) and forwarding API requests to the backend.
- **Monitoring and Logging**: The **EK Stack** (Elasticsearch, and Kibana) is used for **real-time monitoring**. This setup allows tracking **system health** and **maintaining performance** visibility in a production environment.
- **Sensors**: The system ingests data from virtual sensors designed to simulate real-world data, feeding it into Kafka for processing. This layer generates data-driven insights, which are then made accessible to users through the frontend.
- **Docker**: The entire application is containerized using **Docker**, ensuring portability, ease of deployment, and scalability across different environments.

This architecture efficiently separates concerns, supports high performance, and allows for future scalability, making it ideal for applications that need real-time data processing, analytics, and a responsive user interface.

Table 1. Technologies and Their Roles in the Project

Technology	Purpose in Project
Nginx	- Reverse Proxy - Monitoring - Caching
Spring (Spring Boot)	- API & Business Logic Layer - Integration with Databases - Message Handling - Integration with Kafka

Kafka	- Asynchronous Task Processing - Message Broker - Event-Driven Architecture
PostgreSQL	- Relational Data Storage - Advanced Querying - Spring Integration
TimescaleDB	- Time-Series Data Storage - Efficient Time-Series Querying - Scalability & Retention Policies - Seamless PostgreSQL Integration
Docker	- Containerization - Portability & Scalability - Efficient Resource Management
EK Stack	- Log Aggregation - Real-Time Monitoring
React	- Frontend Framework - Integration with APIs
Vite	- Development Build Tool - Fast Development Server
Tailwind CSS	- Pre-build utility classes to style elements - Responsive Design

Module interactions

*1. **Sensor Data Flow***

- **Physical Sensors → Kafka Producer**: Physical sensors collect data (e.g., heart rate, body temperature and respiratory rate) and send it to the **Kafka Producer**.
- **Kafka Producer → Kafka Broker**: The Kafka Producer converts the incoming sensor data into Kafka records and sends them to the **Kafka Broker** for distribution.

*2. **Data Distribution***

- **Kafka Broker → Data Processing Layer**:
  - The Kafka Broker stores the sensor data in **Kafka Topics**, and the **Data Ingestion Module** in the Data Processing Layer subscribes to the relevant Kafka topics.
  - The **Data Aggregation Module** and **Data Transformation Module** process the data to create meaningful outputs, such as calculating averages or transforming raw sensor data into standardized formats.

- **Persistence Logic Layer → Relational/Time-Series Database**: Processed data can be stored in both relational and time-series databases. This ensures historical sensor data is available for querying later or for generating analytics.
- **Logs Analysis Layer → Elasticsearch**: Data is also forwarded to Elasticsearch, where it can be used for real-time search and analytics. Elasticsearch indexes the data for fast retrieval and advanced querying through **Kibana**.

3. Business Logic Layer

- **Team Service, Player Service, Session Service, User Service, Auth Service, Security Service, and Administrator Service**: These microservices in the **Business Logic Layer** handle various app-specific processes. For example, the **Player Service** handles user-related data and tracking.
- **Administrator Service**: Allows the system administrator to configure settings and manage system-wide functions, ensuring the overall system runs smoothly.

4. Analysis Layer

- **Real-Time Analytics**: Using the real-time sensor data, the system can compute analytics in real-time.
- **Historical Analytics**: Stored data in the relational/time-series databases or Elasticsearch is used for generating historical insights.

5. User Layer

- **REST API → Web Applications**: The REST API allows mobile and web applications to query data, request analytics, and retrieve user-specific information (e.g., user session history, sensor data trends).
- WebSocket **→ Web Applications**: WebSocket plays a critical role in ensuring real-time data ingestion, delivering live updates seamlessly within the application.

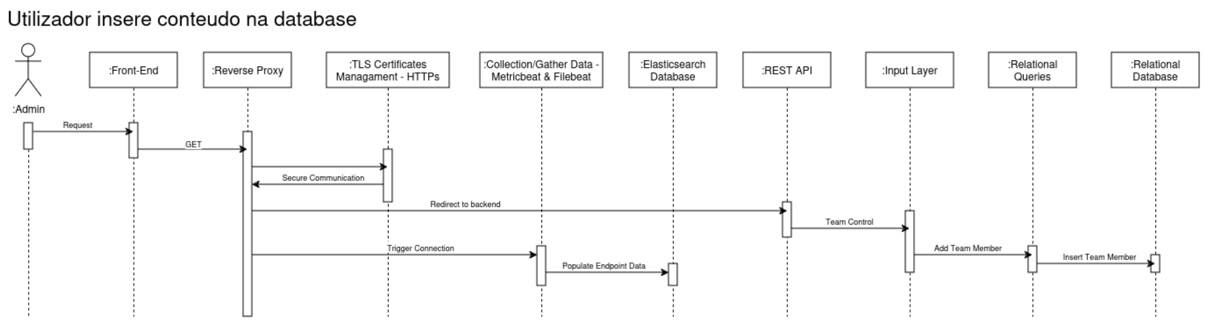


Figure 7. Sequential Diagram: User Content Insertion into the Database

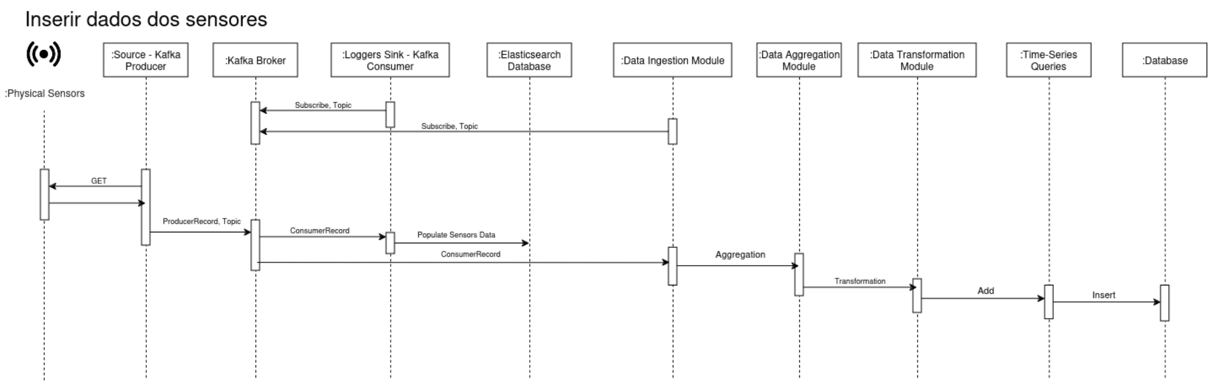


Figure 8. Sequential Diagram: Sensor Data Ingestion and Processing Workflow

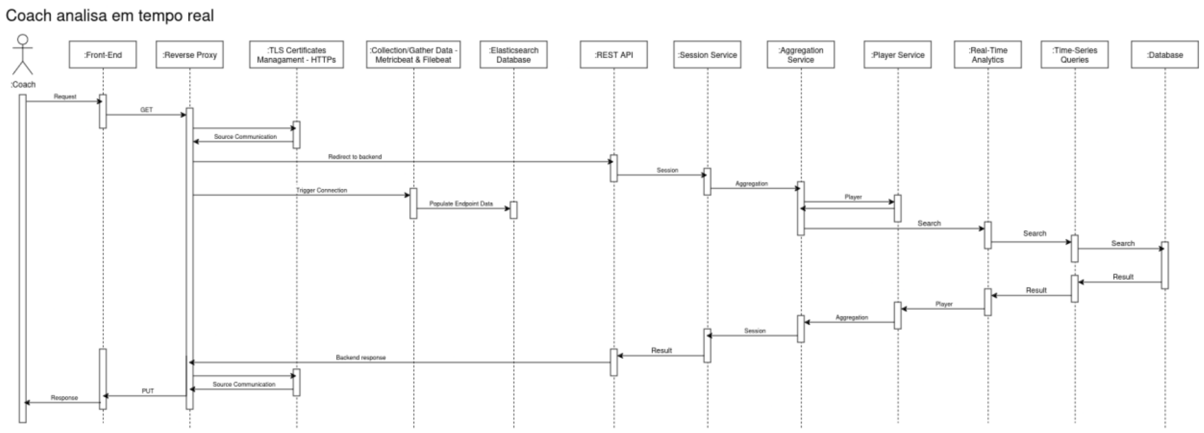


Figure 9. Sequential Diagram: Real-Time Analysis by Coach

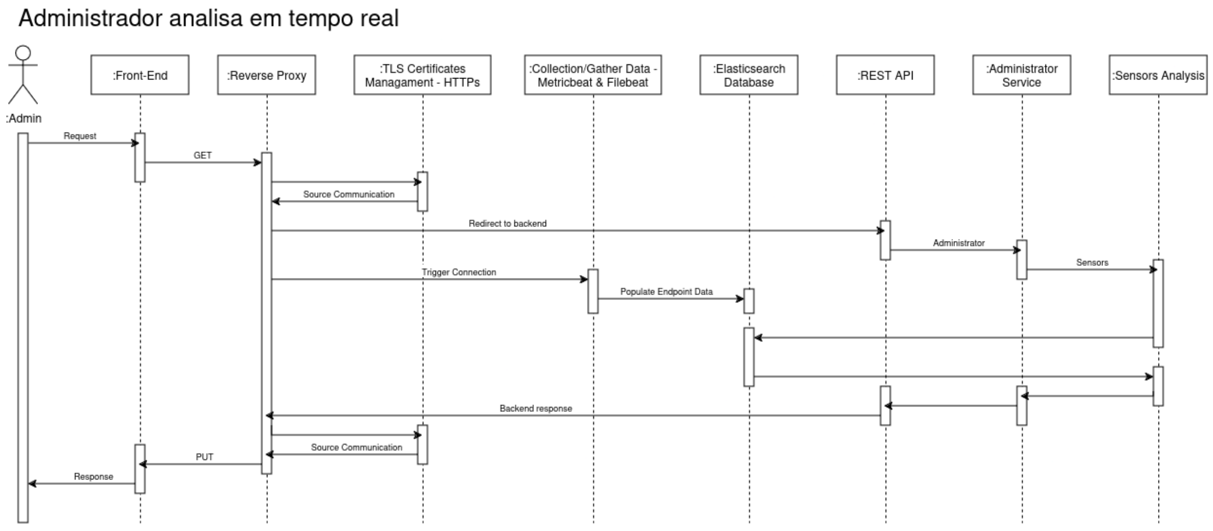
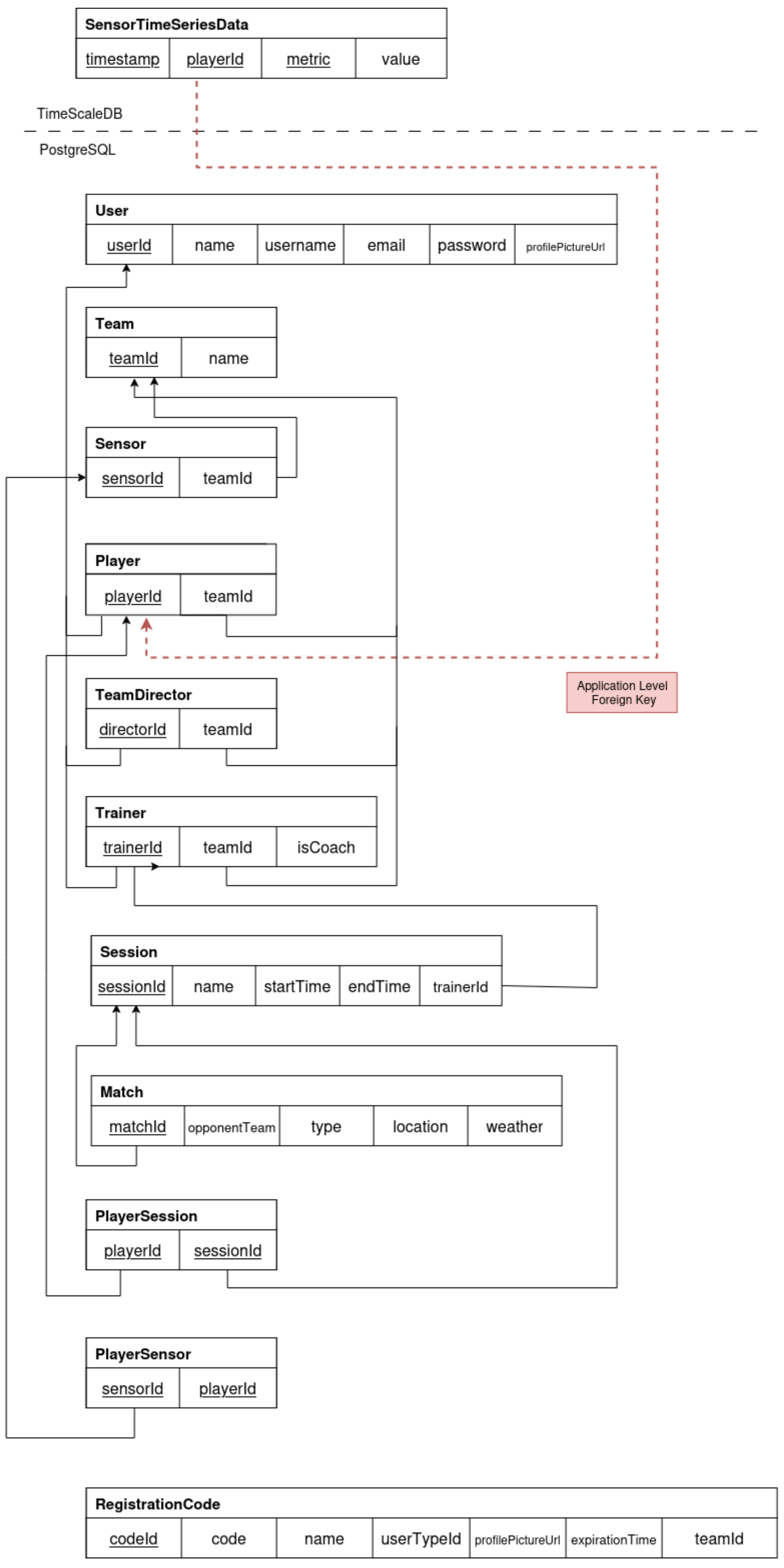


Figure 10. Sequential Diagram: Real-Time Analysis by Admin

4 Information perspective

This section focuses on the key concepts managed within the domain and their relationships. The application handles various entities such as Users, Teams, Sessions, Matches, and Sensor Data, each with their respective attributes and connections. These concepts are modeled using a logical representation of the domain in the form of ER diagram to showcase the data structure, relationships, and persistence layer.

The diagram below illustrates the logical data model. It highlights the integration of a relational database (PostgreSQL) for user, team, and session management, alongside a time-series database (TimeScaleDB) for handling sensor data. The relationships between the entities are explicitly defined, ensuring a clear understanding of the connections in the system.


SensorTimeSeriesData
timestamp
playerId
metric
value
TimeScaleDB
PostgreSQL
User
userId
name
username
email
password
profilePictureUrl
Team
teamId
name
Sensor
sensorId
teamId
Player
playerId
teamId
Application Level Foreign Key
TeamDirector
directorId
teamId
Trainer
trainerId
teamId
isCoach
Session
sessionId
name
startTime
endTime
trainerId
Match
matchId
opponentTeam
type
location
weather
PlayerSession
playerId
sessionId
PlayerSensor
sensorId
playerId
RegistrationCode
codeId
code
name
userTypeId
profilePictureUrl
expirationTime
teamId

5 References and resources

In developing the Smart Training System, several key technologies and resources were instrumental. Below is a curated list of these components, including their official websites:

- **TimescaleDB**: An open-source time-series database optimized for fast ingest and complex queries, built on PostgreSQL. Official website: https://www.timescale.com/
- **Apache Kafka**: A distributed event streaming platform capable of handling millions of events a day, used for building real-time data pipelines and streaming applications. Official website: https://kafka.apache.org/
- **Kafdrop**: A web UI for viewing Kafka topics and browsing consumer groups, facilitating easier management of Kafka clusters. GitHub repository: https://github.com/obsidiandynamics/kafdrop
- **Apache ZooKeeper**: A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Official website: https://zookeeper.apache.org/
- **Elasticsearch**: A distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. Official website: https://www.elastic.co/elasticsearch
- **Kibana**: A data visualization and exploration tool for reviewing logs and time-stamped data stored in Elasticsearch. Official website: https://www.elastic.co/kibana
- **Spring Framework**: A comprehensive framework for enterprise Java development, providing infrastructure support for developing Java applications. Official website: https://spring.io/
- **Vite**: A frontend build tool that offers a faster and leaner development experience for modern web projects. Official website: https://vitejs.dev/

These resources were invaluable in the development of the Smart Training System and can serve as a guide for others undertaking similar projects.