Minimum Viable Product (MVP)

Project Management Laboratory

21st March, 2025
LGP-17

António Augusto Brito de Sousa, 202000705
Daniel Cabral Bernardo, 202108667
Gustavo Nuno Ferreira Tarábbia, 202104655
Leonor Silva Santos de Azevedo Maia, up202302864
Pedro de Almeida Lima, 202108806
Rodrigo Campos Rodrigues, 202108847
Rodrigo José de Castro Pinheiro, 202403040
Tomás Alexandre Soeiro Vicente, 202108717
Tomás Eiras Silva Martins, 202108776

Index

1. Requirements

1.1. Developed Requirements

- **Functional Requirements**

Code	Rule	Description	Status
FR01	Intuitive touch-based input system	The interface must follow established touch UI/UX patterns with consistent button placement and sizing appropriate for fast-paced game environments. The system must provide tactile and visual feedback to confirm successful data entry. Sport-specific templates must automatically organize input controls relevant to the selected sport. Input areas must be sized and spaced to minimize accidental entries during high-pressure game situations.	**Fully implemented**
FR02	Game setup wizard for pre-game configuration	The user must have access to a pre-game setup wizard so that the game information can be validated and updated by the user.	**Partially Implemented** (only player creation)
FR03	Login	The user responsible for the scoreboard must have an account which will then give them access to the games they are responsible for.	**Fully Implemented**
FR04	Main Page	The system must have a main page that allows users to search for a specific game. To facilitate this process, the page must display different filters (sports, leagues, time of the day, court). In addition, a settings button must be accessible.	**Fully Implemented**
FR05	Board Alert	The board shall provide visual alerts for critical game events (e.g., goals, fouls, timeouts, game end).	**Partially Implemented** (information updates, but doesn't alert

			everything)
FR06	Multi-User Collaboration on the Table	The system shall allow multiple users to simultaneously operate the control panel from different devices or browser tabs. Each user can have specific responsibilities, such as one managing fouls while another controls the game clock. All updates, deletions, and new game events shall be synchronized in real-time across all connected devices, ensuring seamless coordination and accurate game management.	**Fully Implemented**
FR07	Manual Override of Any Value	Users must be able to manually change any game value (e.g., score, fouls, penalties) at any time without restriction.	**Fully Implemented**
FR08	Time Control	The system must allow users to start, stop, pause, and rewind the game clock and other timers (timeouts, ball possession timers).	**Fully Implemented**
FR09	Log Acess	A detailed history log should be available for review and accountability.	**Partially Implemented** (unknown author)

Table 1 - Functional requirements

- **Non-Functional Requirements**

Code	Rule	Description	Status
NF01	Availability	The system must be available 99 percent of the time every day.	**Undefined** (as there never was a live host server)
NF02	Performance	The system must display updates within 500ms of input and handle a rapid succession of events without lag.	**In Agreement**
NF03	Data integrity	The system must ensure the maintenance and assurance of data accuracy and consistency.	**In Agreement**
NF04	Usability and Compatibility	Interface learnable within 10 minutes for new officials. The system must be compatible with modern browsers, including Chrome, Firefox, and Edge. The control panel interface must be optimized for 10” tablets or larger.	**In Agreement**

		The system must be responsive and work correctly on mobile devices and desktops.	
NF05	Security	Only authorized users should be able to edit game events and statistics.	**In Agreement**
NF06	Scalability	The system must allow for easy addition of new sports without the need for major code modifications. The database must be designed to support a large growth in the number of games recorded over the next years. System updates must be applied without impacting ongoing matches.	**In Agreement**
NF07	Maintenance	The source code must follow development standards (E.g. design patterns) to facilitate maintenance and have proper documentation.	**Partly in agreement** (even though the code isn't completely compliant with design patterns, it's organized)
NF08	Compliance with Sports Standards	The system must follow the official rules and standards of each sport (handball, basketball, volleyball, hockey, and futsal…) to ensure the accuracy and validity of the information displayed.	**Partly in agreement** (different rules for games in different settings (cup vs league))

Table 2 - Non-Functional requirements

1.2. User Stories

User Story	Priority	Status
As a scoreboard operator, I want to log into my account, so that I can be authenticated.	**Must Have**	**Fully Implemented**

As a scoreboard operator, I want a guided pre-game setup wizard, so that I can validate and update game information before the match starts.	**Must Have**	**Partially Implemented** (only player creation)
As a scoreboard operator, I want to see a list of games I am assigned, so that I can easily find and select the game I need to officiate.	**Must Have**	**Partially Implemented** (no filter for assigned games)
As a scoreboard operator, I want to view the details of a game before the match starts (teams, time, location, other referees) so that I am prepared and informed.	**Must Have**	**Fully Implemented**
As a scoreboard operator, I want to have a clear indication of my role in a game if there are multiple people assigned to operate the scoreboard, so that we can coordinate our responsibilities.	**Could Have**	**Not Implemented**
As a scoreboard operator, I want my officiating team to be able to access and update the same game scoreboard from different devices simultaneously, allowing each official to manage specific aspects of the game while maintaining a single, synchronized scoreboard display.	**Should Have**	**Fully Implemented**
As a scoreboard Operator, I want a main page with search and filtering options, so that I can quickly find and access a specific game.	**Should Have**	**Fully Implemented**
As a scoreboard operator, I want to add and edit goals/points, so that I can ensure the correct score is displayed when points are scored or scoring decisions are reversed.	**Must Have**	**Fully Implemented**
As a scoreboard operator, I want to record and modify player cards depending on the game rules, so that I can accurately track disciplinary actions and correct any recording errors.	**Must Have**	**Fully Implemented**
As a scoreboard operator, I want to log and edit team and individual fouls, so that the correct foul count is maintained throughout the game.	**Must Have**	**Fully Implemented**
As a scoreboard operator, I want to record and edit team timeouts, so that I can track remaining timeouts for each team and fix any incorrect entries.	**Must Have**	**Fully Implemented**
As a scoreboard Operator, I want to be able to view and edit a detailed history log, so that I can track changes.	**Should Have**	**Fully Implemented**

As a scoreboard Operator, I want the changes I introduced in the table to be shown in realtime in the scoreboard.	**Must Have**	**Fully Implemented**
As a scoreboard Operator, I want to be able to check and edit, if necessary, the game report after the match ends, including scores, penalties, and any relevant notes.	**Should Have**	**Not Implemented**
As a scoreboard Operator, I want the app to work reliably and offline to a certain extent (e.g., basic game data cached), so that I can use it even if the internet connection is unstable in the venue.	**Could Have**	**Not Implemented**
As a scoreboard Operator, I want to have a small indicator that alerts me if there are any connection issues between the app and the display screen, so that I can troubleshoot quickly.	**Should Have**	**Not Implemented**
As a scoreboard Operator, I want an adapted interface so it fits the matches' needs.	**Should Have**	**Fully Implemented**
As a scoreboard Operator, I want to start, pause, and update (rewind or forward) the game clock, so that the game time can be accurately managed.	**Must Have**	**Fully Implemented**
As a scoreboard Operator, I want to identify the scoring player, so that goals are accurately attributed.	**Must Have**	**Fully Implemented**
As a volleyball scoreboard Operator, I want to signal Ace points, so the game is more dynamic, by showing a pop-up with the information and the player's image.	**Could Have**	**Not Implemented**
As a volleyball scoreboard Operator, I want to be able to manage sets and track the current set score, so that the game progress is clearly displayed.	**Must Have**	**Fully Implemented**
As a volleyball and basketball scoreboard Operator, I want to be able to manage substitutions and potentially record player information if needed for game reports, so that player changes are tracked.	**Must Have**	**Fully Implemented** (basketball substitutions were removed from scope)
As a futsal scoreboard Operator, I want to track accumulated fouls for each team, so that penalty situations are correctly identified.	**Must Have**	**Partially Implemented** (no warning, but identified)

As a futsal scoreboard Operator, I want to be able to issue cards to players, so that disciplinary actions are recorded.	**Must Have**	**Fully Implemented**
As a futsal scoreboard Operator, I want to be able to manage penalty kicks and record the outcome (goal, miss, save), so that penalty situations are handled correctly.	**Could Have**	**Not Implemented**
As a Spectator, I want the scoreboard to display visual alerts for critical game events so that I can quickly recognize key moments of the game.	**Could Have**	**Partially Implemented** (Just for substitutions)
As a Spectator, I want the scoreboard to be easily readable from anywhere in the venue, so I can follow the game without straining.	**Should Have**	**In Agreement**
As a Spectator, I want to have a final view of the scoreboard when the game ends, that shows-off the winner and the most relevant events in the game.	**Could Have**	**Not Implemented**

Table 3 - User Stories

2. Architecture

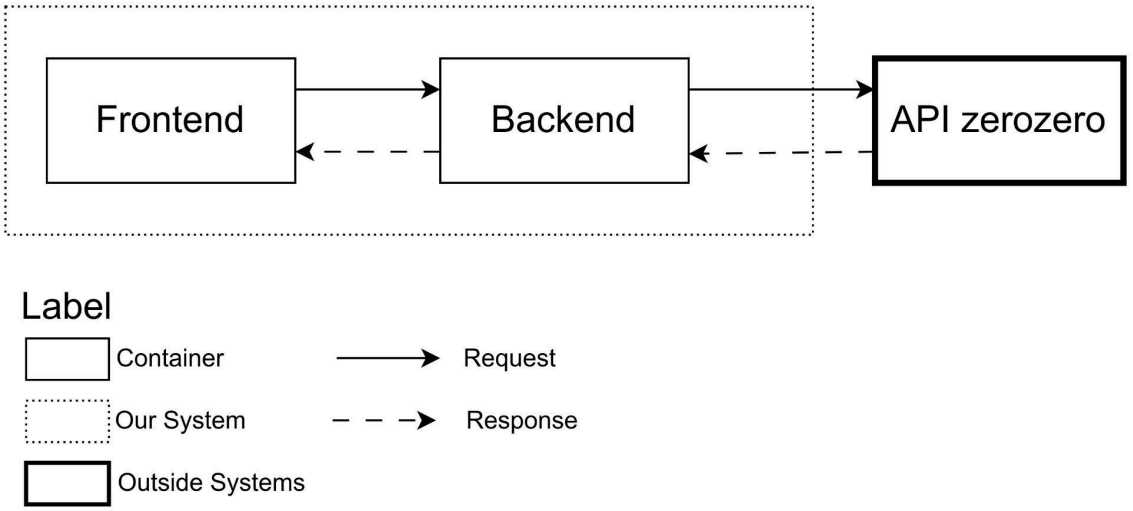


Figure 1 - Containers diagram

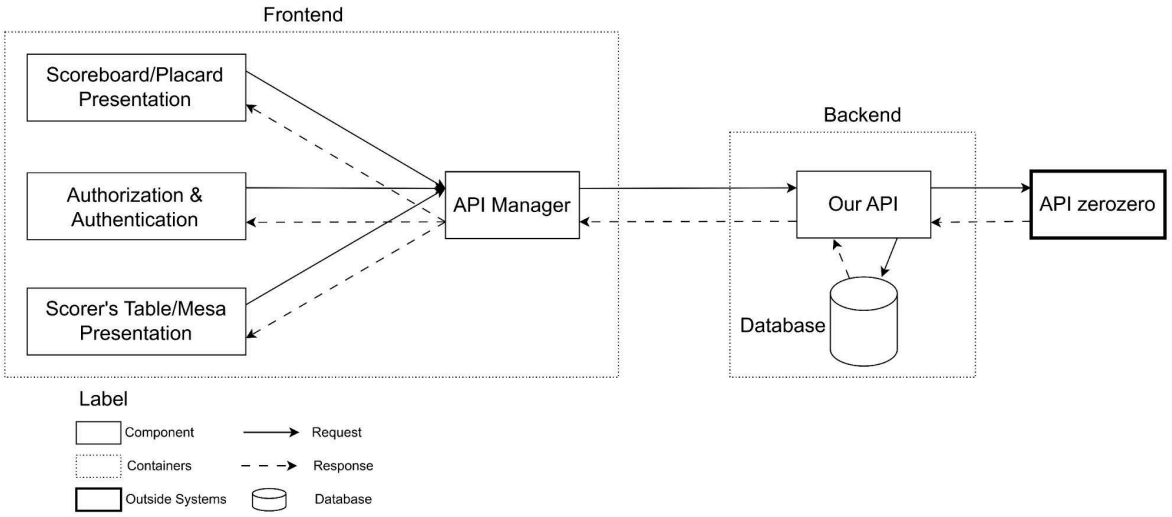


Figure 2 - Components diagram

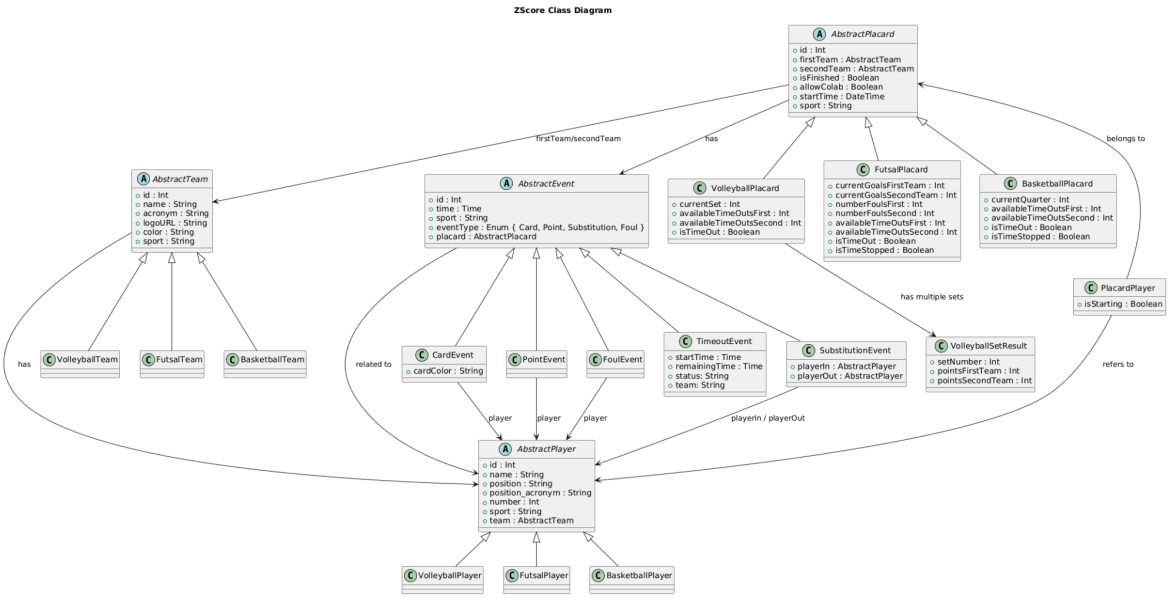


Figure 3 - Class diagram

The ZScore system employs a client-server architecture where a central server acts as the core component, retrieving pre-game information (teams, players, etc.) from the external zerozero API and maintaining persistent data storage through a connected database.

The application presents two distinct user interfaces: the “Placard" view, which functions as the scoreboard visible to spectators, and "Mesa" views operated by scoring officials at the scorer's table. Although the diagrams only have one frontend, it is possible to have various frontend instances (various scoreboards and various scorer’s tables) active at the same time, and all of them will be synchronized. During gameplay, officials interact with the scorer’s table interfaces to record events in real time, with these inputs transmitted to the central server for processing and database storage.

Both the scoreboard presentation and the scorer’s table presentation interact with the Api Manager, which is responsible for making requests to the backend and hides this process from the presentation components.

All application instances, including both the scoreboard and other scorer’s table views, update their information by polling the central server at regular intervals to receive the latest game state. This polling mechanism ensures that as scoring officials document game events through their user-friendly interfaces, the information is eventually reflected on the scoreboard display and synchronized across all scorer’s table instances while maintaining data integrity and enabling efficient distribution of scoring responsibilities among multiple officials.

Anyone can access the placard, but only authorized game officials can access the scorer’s Table, using the authorization and authentication component.

We also created a class diagram to model our data and guide the implementation of the frontend, backend and database. It reflects the multiple types of games, teams, players and events.

We deliberately, moved as much computation as possible to the frontend, because, in the future, we expect a lot of clients to interact with our servers at same time, and thus, decrease the amount of data sent/received through the network and decrease the workload on servers, making the cost of running the system lower.

The streamlined workflow supports the system's primary goal: enabling quick decision-making and accurate event registration at the scorer's table while providing spectators with near real-time visual updates.

3. Technologies

For practical development, the following technologies were implemented in the project:

- **React**
  ReactJS is a component-based JavaScript library we used to build a dynamic and interactive user interface. The frontend is crucial as most computations are handled on the client side, ensuring a smooth user experience. React's component architecture was particularly advantageous for our sports scoring system, as it enabled efficient reuse and customization across different sports contexts. While various sports required unique interface elements and functionality, many core components could be designed once and repurposed with sport-specific modifications. This approach significantly reduced development time and maintenance overhead while maintaining a consistent user experience and visual identity across all implementations.

- **Vite**
  Vite is a fast development tool for React and other modern frontend frameworks. It quickly compiled the code and updated the browser instantly during the development phase. It also included a built-in server for serving the frontend and supported fast reloading, which made development smoother and more efficient.

- **PHP**
  PHP is a server-side scripting language designed for web development. Most of our backend was built using PHP, which was responsible for setting up the REST API. The project utilized PHP-FPM as seen in the backend Docker setup. There were multiple technologies that could have fit this purpose, but PHP was one that we were more comfortable with and was the one that was easier for the client (zerozero) to expand upon.

- **MariaDB**
  MariaDB is an open-source relational database. We used MariaDB to store and

manage all the data from the games. Also, even after the game ended, zerozero could use this stored data in their website. There were also multiple technologies that could have fit this purpose, but MariaDB was the one that was easier for the client to expand upon.

- **REST API**

  A REST API is a web service that allows communication between client and server using HTTP requests. The first time a placard connected to the server, it asked for data using a REST API. This was the easiest way to communicate between the frontend and backend and it had become the industry's standard for this. The PHP backend served via NGINX was set up to handle API requests.

- **Polling**

  Polling is a technique where the client periodically sends requests to the server to check for new updates. This meant that, in this case, both the placard display and the table repeatedly requested the latest game information at set intervals. This approach was straightforward to implement using JavaScript, as it relied on simple HTTP requests through fetch() or XMLHttpRequest within a setInterval() function. Polling kept the system updated without persistent connections but could cause unnecessary network traffic and slight delays. It was a useful initial step.

- **NGINX**

  NGINX is a high-performance web server and reverse proxy. We used it to serve the PHP backend and handle REST API requests. It also ensured fast and secure communication (though HTTPS setup would have been part of a full deployment).

- **Redis**

  Redis is an in-memory key-value data store integrated into this project to efficiently manage real-time game data such as scores, fouls, cards, substitutions, timeouts, and timer states, ensuring fast and atomic updates essential for a responsive scoreboard application. We have chosen it for its high speed, ease of use, and support for atomic transactions, Redis enables the backend to handle multiple rapid

data operations, such as incrementing scores, tracking timeouts, and updating game events, all while maintaining consistency and minimizing latency.

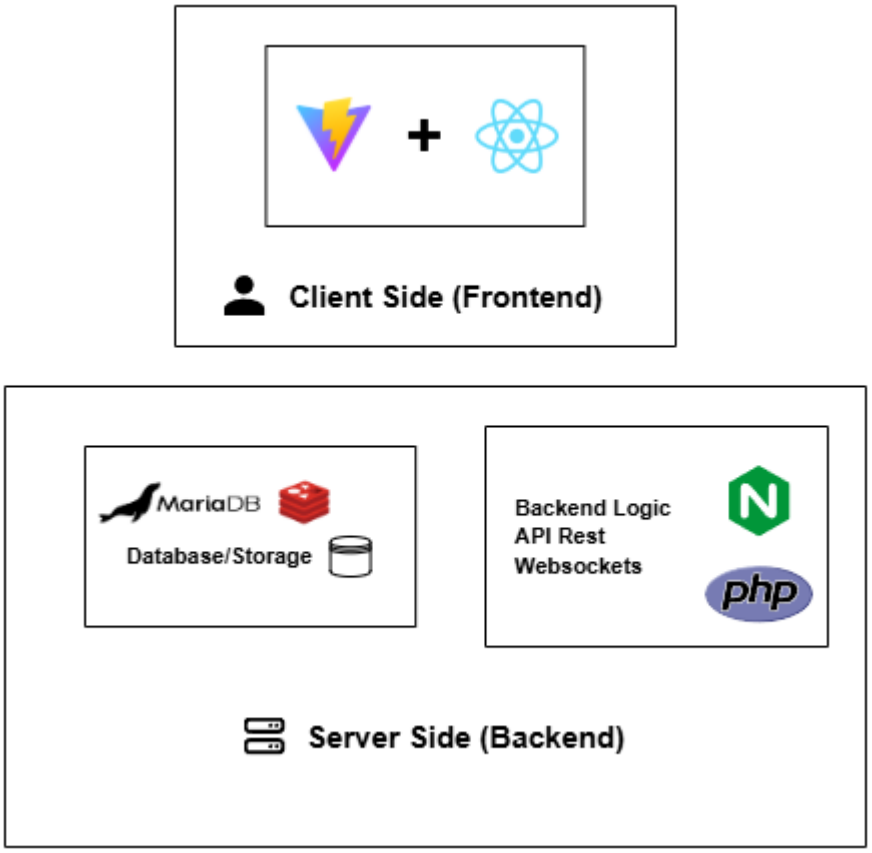


Figure 4 - Tech Stack Diagram

4. Hand-Over

Deliverables

- **Code Repository**
  A complete repository containing the full source code and its development history.

- **Feedback Report**
  A document summarizing the insights collected from three interviews conducted with professional game officials.
  Each official participated in an individual interview session, during which they were asked to use the product and share their impressions, suggestions, and concerns. The discussions focused on the product's usability, usefulness in real-world scenarios, and any features they believed were missing or could be improved.
  While the interviewed officials expressed that the product was very good, they also provided numerous suggestions for improvement, including potential new features.
  These interviews were recorded, so based on these recordings, all their comments and suggestions will be compiled into the feedback report.

- **Future Work Document**
  A detailed list of all identified areas for improvement and possible new features, including all known limitations and enhancements we have considered during development.

- **Technical Documentation**
  Full documentation covering the codebase, system architecture, and design choices.

- **API Documentation**
  A comprehensive reference of all API endpoints, including usage instructions and parameters.

Process

- **Repository Setup**
  A new repository will be created to host the finalized version of the project. All code, including its history, will be migrated to this new repository.
  The current repository was created by FEUP, so to make the zerozero team more comfortable with it, we will create a new one.

- **Handover Meeting**
  A meeting will be held with the next team responsible for continuing the development and maintenance of the product.
  During this session, we will:

    - Present the feedback from the game officials

- Discuss future work
- Provide access to the code repository and all related documentation

5. Administrator and User Manuals

The user manual serves as a description of the user experience and core functionality of ZScore. It provides a step-by-step guide on how to interact with the application, ensuring easy access to key features like accessing the scorer's table or the scoreboard.

Access

1. Right now, the only option is to run the server locally, so addresses are to be changed in the future.
2. Once the server is running locally, launch the preferred browser.
3. In the address bar you should enter the following address: localhost:3000/gameList

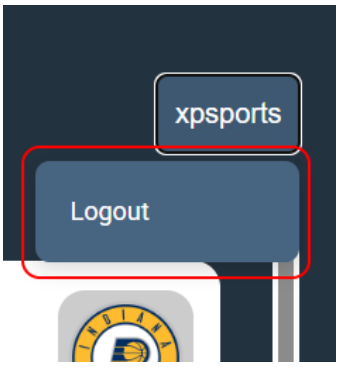
Authentication

- **Login:**
    1. Once you are in the ZScore webpage you should be able to locate a login button in the top right corner.

    2. Upon clicking this button you should be redirected to zerozero's login page where you will enter your account credentials.
    3. If you enter your credentials right, you should then be redirected back to the ZScore game list.

- **Logout:**
    1. After being logged in you are now able to logout by clicking the user button that replaced the login one.

    2. Upon clicking in this button a menu will appear with the logout button.



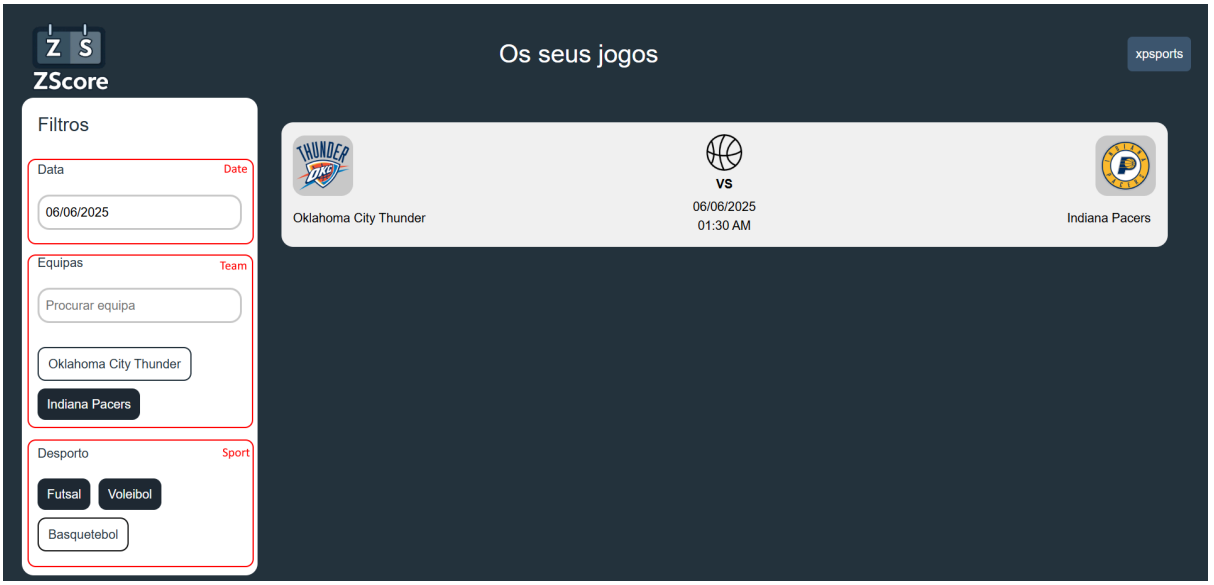
    3. To logout, click the logout button.

Game Search

In order to enhance the usability of the game list, a filter system was introduced. This allows the user to quickly search for the game they desire.

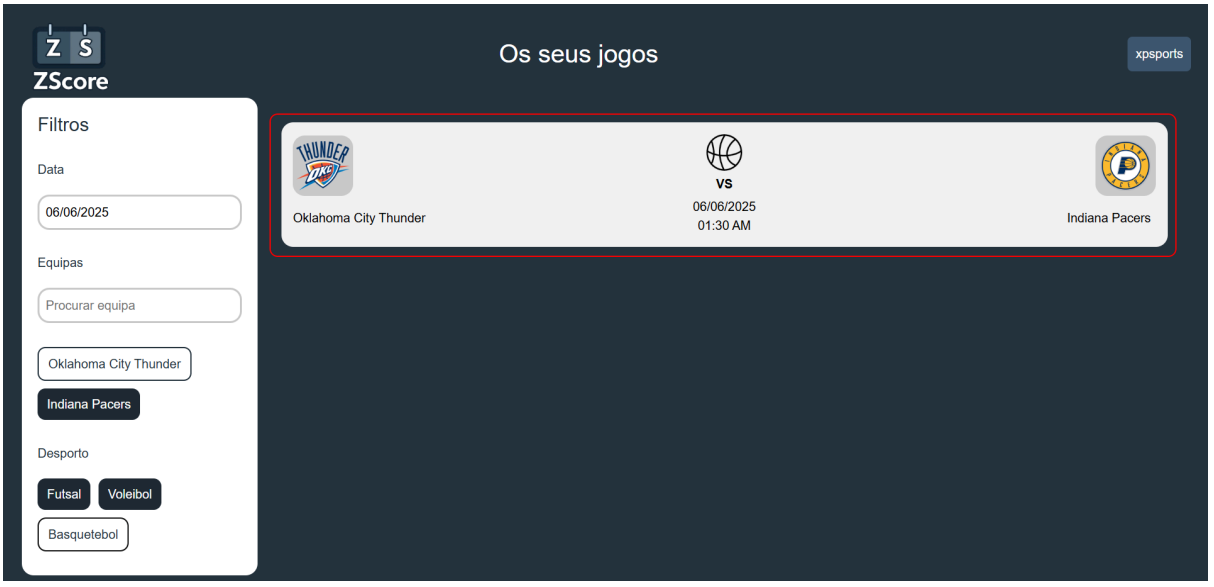
- **Date Filter**
    1. To filter by team, select the date input and select a date.
    2. Optionally, the date can be inserted by typing in the input box
- **Team Filter**
    1. To filter by team, select the team to find.
    2. Optionally, the team's name can be written, in order to find its filter more easily.
- **Sport Filter**
    1. To filter by sport, select the sport to filter by.

These filters can be combined for better search.

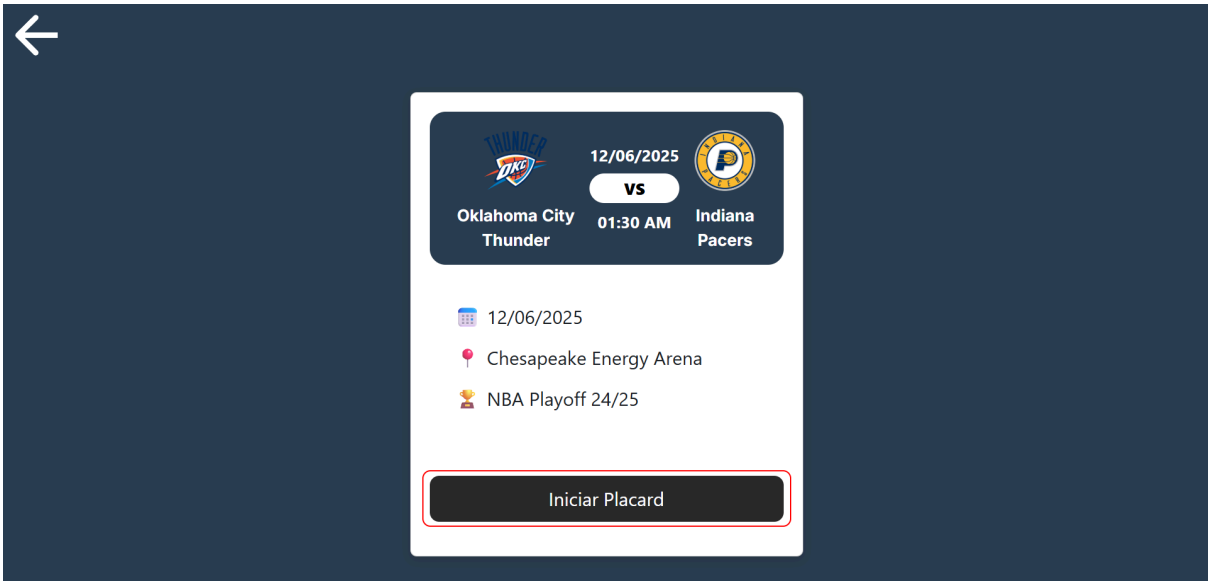


Scoreboard

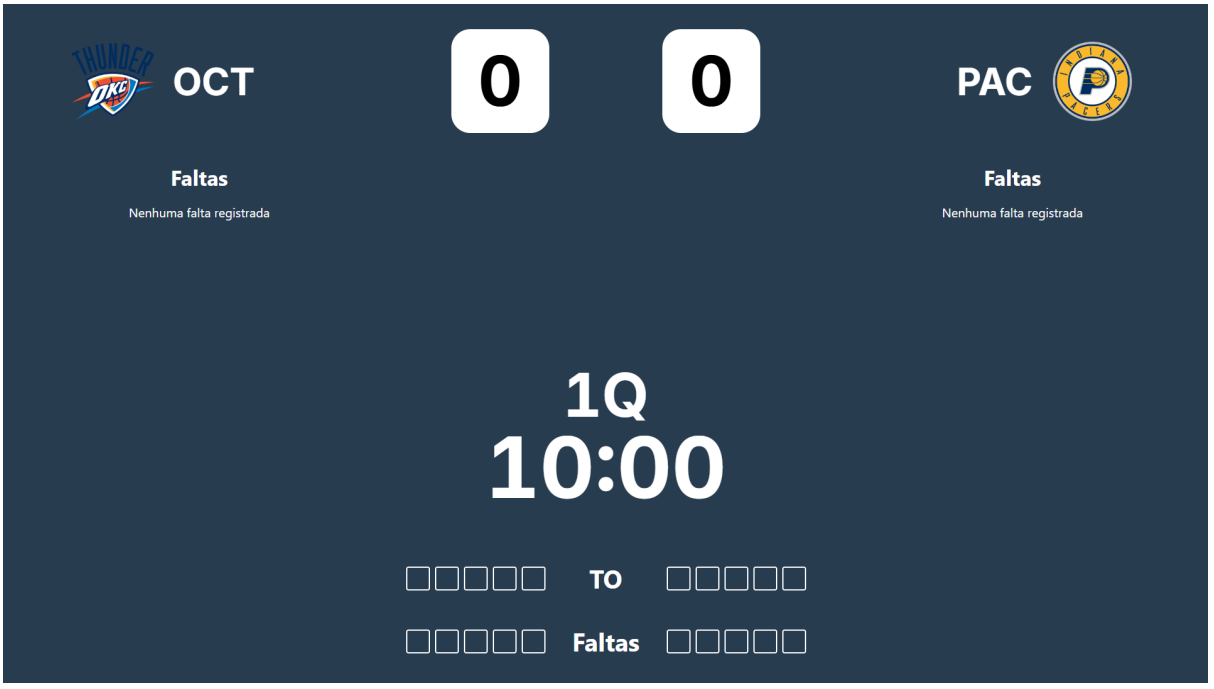
1. Once the user finds the game they want to follow, they should click that game.



2. After that, a menu should appear with game details, on the bottom part of this screen it has a button “Iniciar Placard” - This is the button to enter the game’s scoreboard.



3. Clicking in the previously mentioned button redirects the user for the selected game's scoreboard.



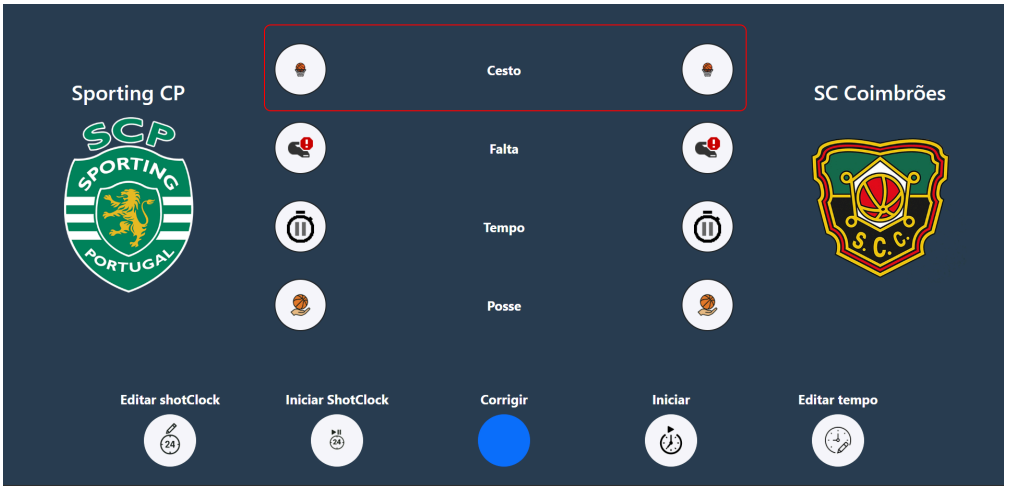
Scorer's Table

1. In order to access a game scorer's table, a user must have collaboration permissions for that game.
2. Upon finding the game the user can collaborate in, they should click that game.
3. Now, a menu appears with game details and two buttons, find the button "Iniciar Mesa" - This is the button to enter the scorer's table
4. If the game has not started yet, then some changes can be made and a pre-game setup page will be prompted. (Explained under "Pre-game Setup")
5. After, the scorer's table is reached.

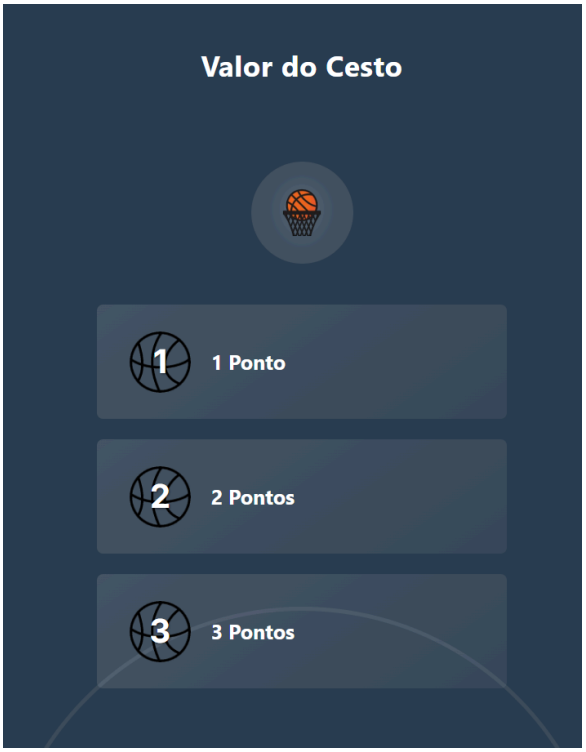
The table counts with varying features depending on the sport being played, as so they are only presented if needed:

Note that for some features there are two buttons, one for each team, so selecting a feature on a team's side means that an event will be registered for that team.

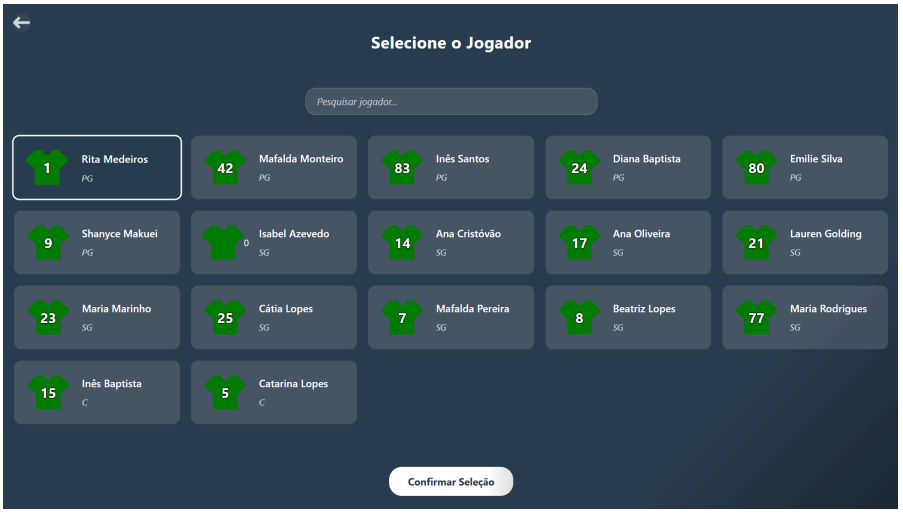
- **Adding goals/points**
  1. To add goals/points, select the score button.



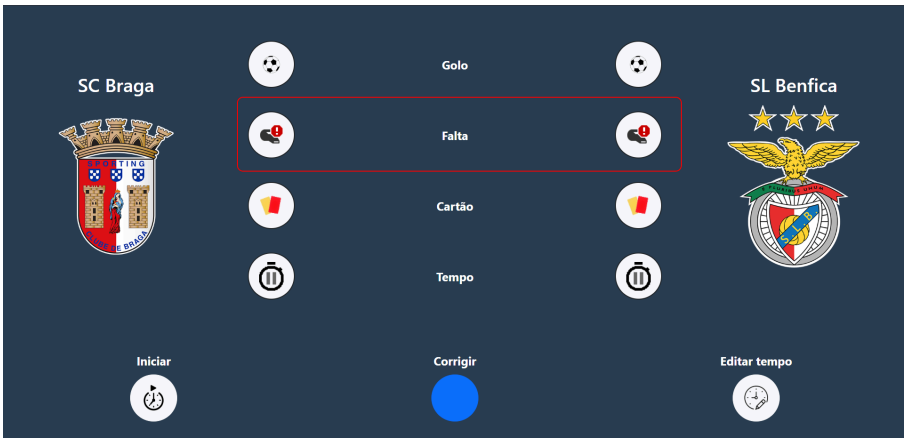
  2. If needed, then the user will be asked to select the number of points to register and then press continue.



  3. Then, also if needed, the user will be prompted to select the player that scored and confirm, returning to the scorer's table main page.



- **Adding fouls**
    1. To add fouls, select the foul button.

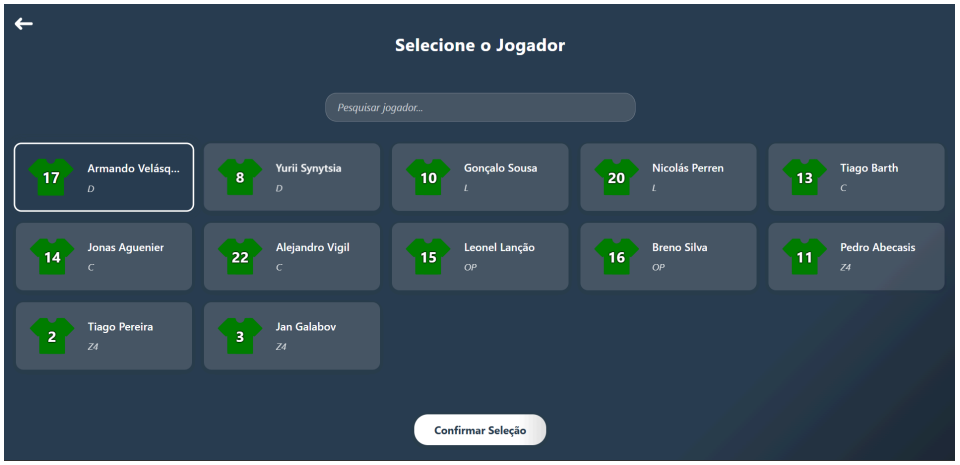


    2. Then, if needed, it will be asked to select the player that committed the foul and continue.

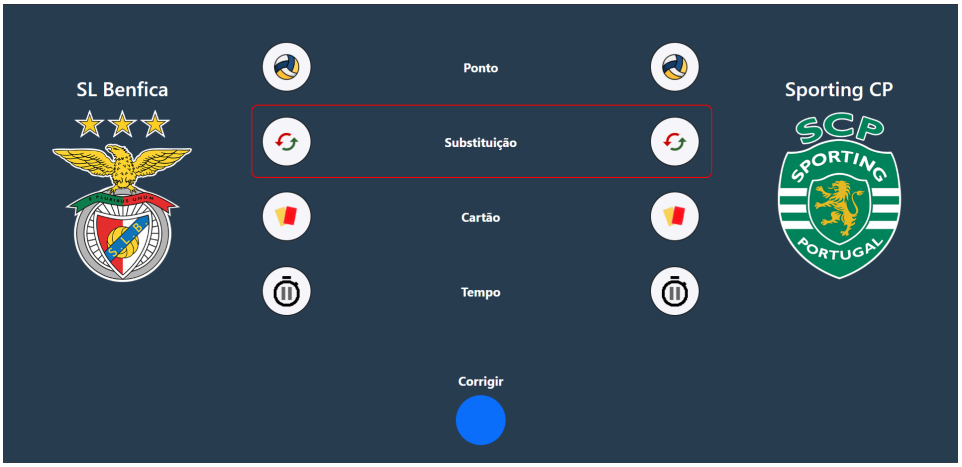
- **Adding cards**
    1. To add cards, select the card button.

2. Then it will be prompted to select the card type and continue.

3. After, to select the player that received the card and continue.



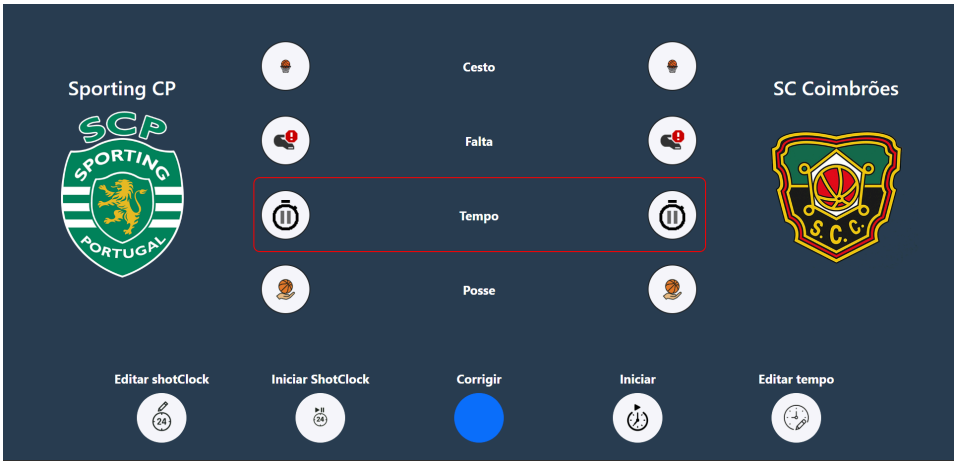
- **Substitution**
    1. To add substitutions, select the substitution button.



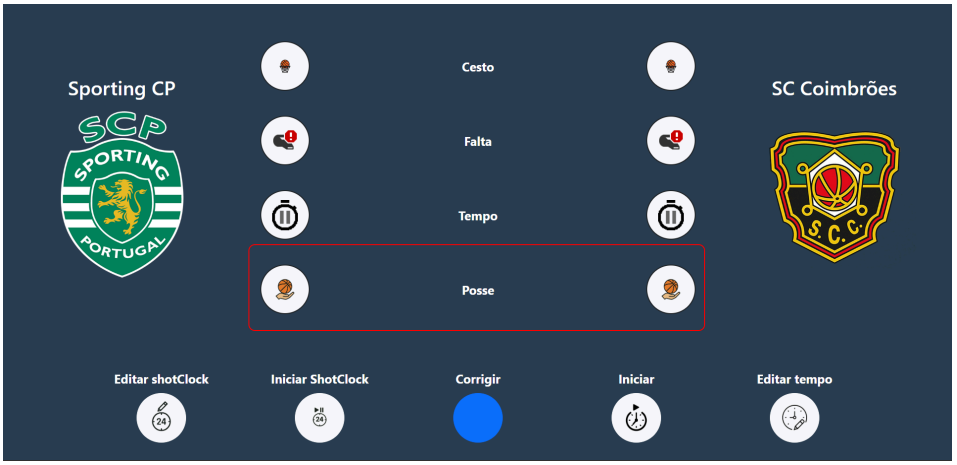
2. Then it will be asked to select the player that is being subbed.

3. After, select the player entering the game and confirm.

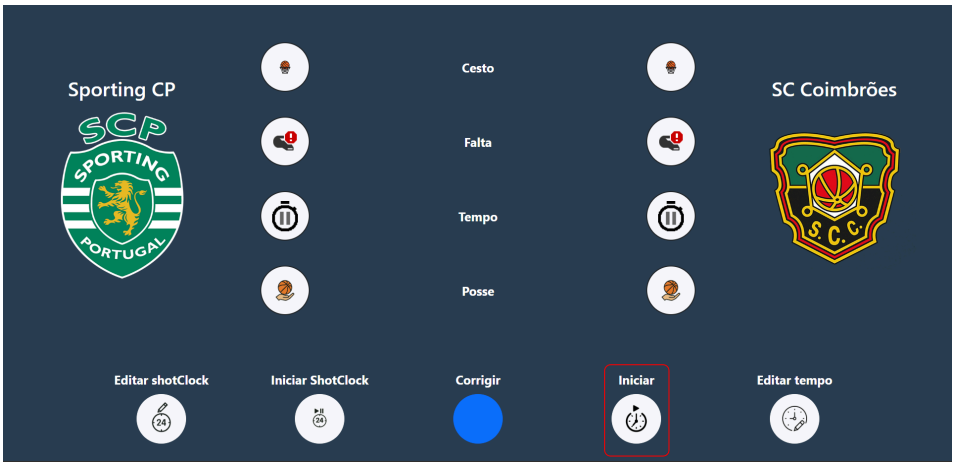
- **Timeouts**
  1. To add timeouts, select the timeout button.



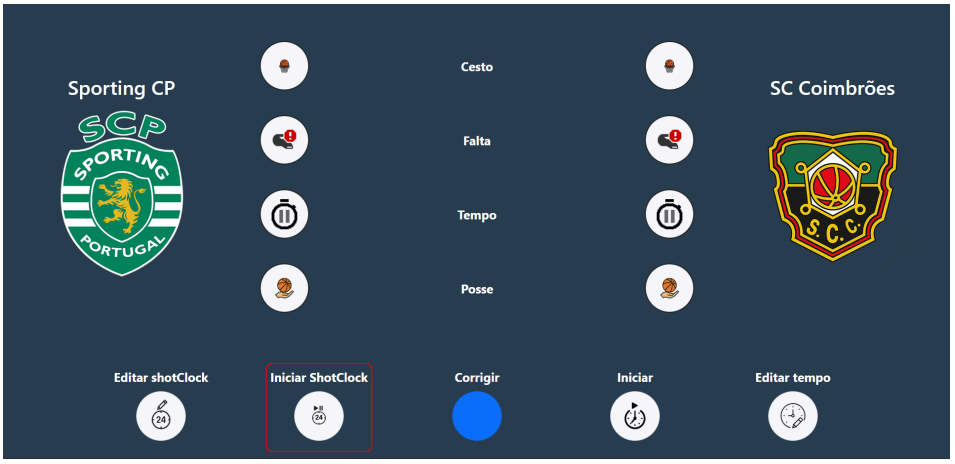
- **Change Ball Possession**
    1. To track ball possession, click the Possession button.



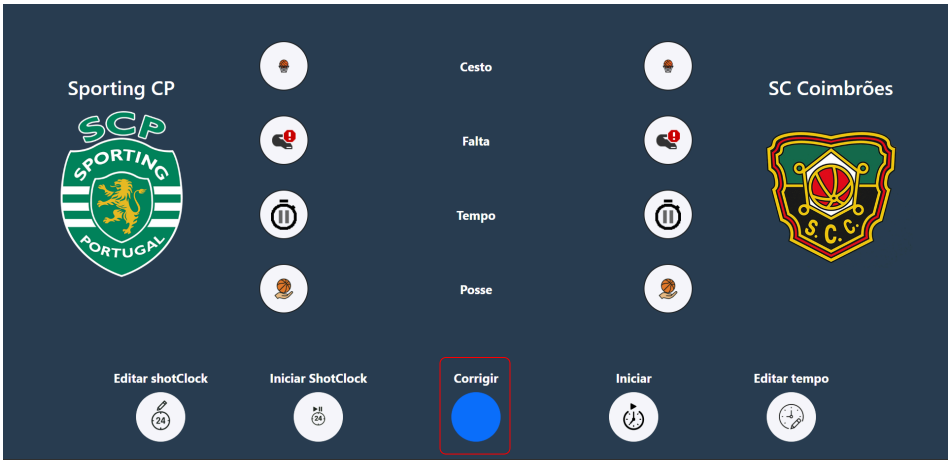
- **Game Clock**
    1. To start/stop the game's clock, click the Start/Stop button



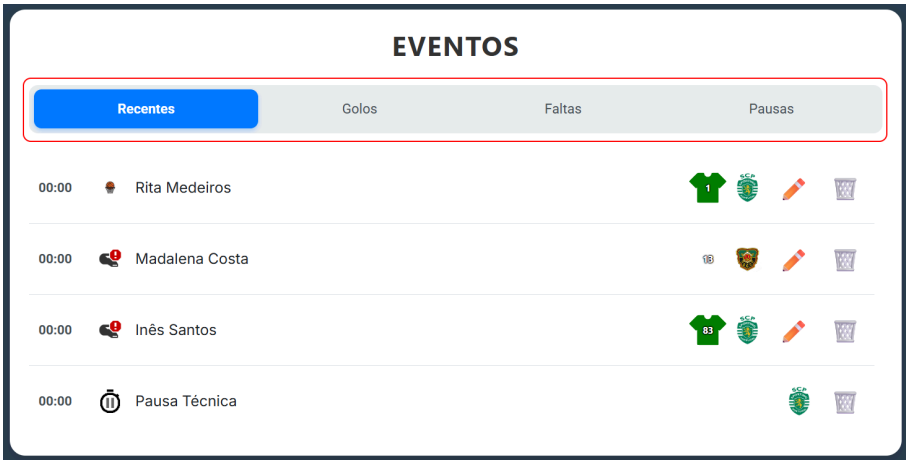
- **Shot Clock**
    1. To start/stop the shot clock, click the shot clock button.



- **Edit Registered Events**
  1. Registered game events can be edited/deleted, for that click the “Corrigir” button.



  2. Now on the events history page, events can be sorted by most recent or by type, select the desired order on the top of the history tab.



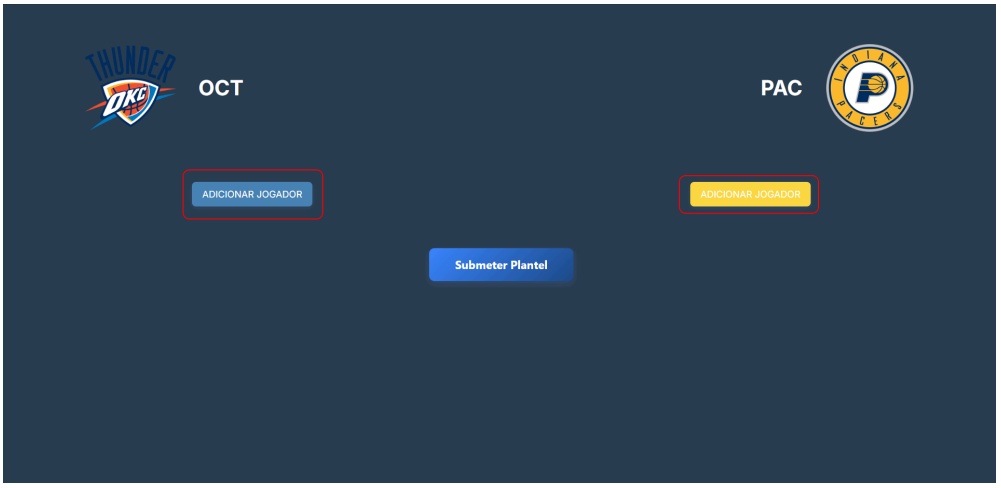
  3. To edit an event, click on the edit button on the desired event and edit the information regarding that event.

4. To delete an event, click on the delete button.

Pre-game Setup

If a game has not started yet, it is possible to edit game information by using the pre-game setup page.

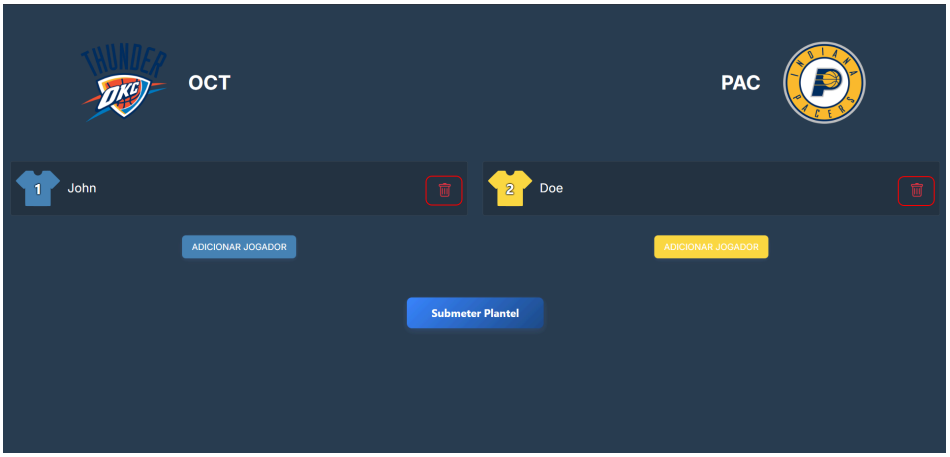
- **Add Players**
  1. In order to add players to a team, there is a button under each team to add players.



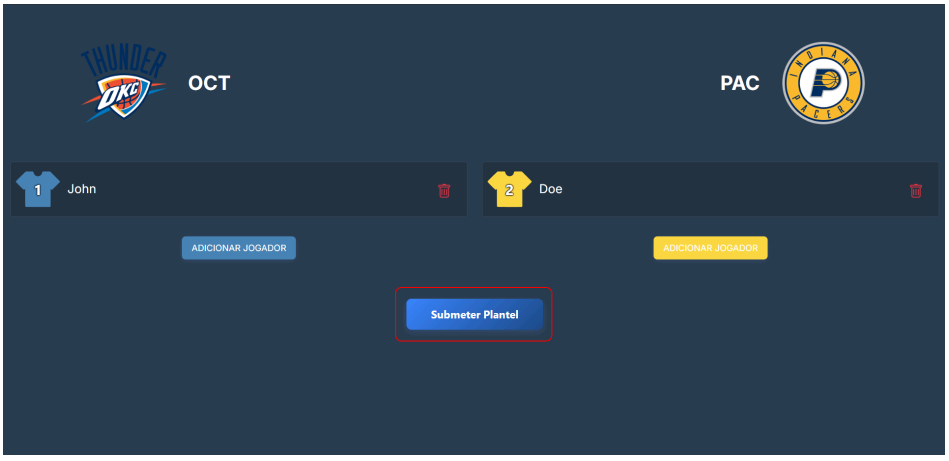
  2. Clicking the button, a prompt appears to insert player info such as name, number and position.

3. Upon entering all the information, confirm and the player will be added to the team lineup.

- **Remove Players**
  1. To remove a player, click the remove button on the player to remove.



After all desired changes have been done, click “Submeter Plantel” in order to get into the scorer’s table.



6. Findings and Metrics

During our Build-Measure-Learn phase, we developed and demonstrated new features to the client on an almost weekly basis. Based on the feedback, we adjusted our focus to align with delivering a product that meets expectations.

Findings and Metrics

1. **Positive Core Usability Validation**:
    - **Finding**: Interviews with professional game officials (as noted in the "Feedback Report") and discussions with the client indicate that the core product is perceived positively. This suggests that the fundamental user interface (UI) and user experience (UX) design for key game operations are on the right track.
    - **Metric (Qualitative)**: Positive verbal feedback and successful task completion by officials during product demonstrations.
2. **Demand for Feature Completeness and Expansion:**
    - **Finding:** While core operations are functional, several requirements critical for a complete user experience are "Partially Implemented" (e.g., FR02 - Game setup wizard beyond player creation, FR05 - Comprehensive Board Alerts, FR09 - Log Access). User stories also reflect this, with several "Must Have" items partially implemented or key "Should Have" items not yet started (e.g., offline capabilities, connection indicators).
3. **Technical Scalability & Real-Time Performance Considerations:**
    - **Finding:** the current polling mechanism for data synchronization, while effective for the MVP, is acknowledged as a potential bottleneck for achieving truly instantaneous updates (NF02 - <500ms target) across many simultaneous users and for future scalability.
    - **Metric (To be tracked)**: Current update latency under test conditions. Future: Network traffic per client, server load under simulated concurrent users. NF02 "Performance" is "In Agreement," suggesting informal validation, but continuous measurement is needed.
4. **Robust Logging and Auditability Needs Improvement:**
    - **Finding:** FR09 "Log Access" is "Partially Implemented (unknown author)." This indicates a potential weakness in traceability, debugging, and accountability. A detailed history log is crucial for dispute resolution and system analysis.
    - **Metric (Qualitative):** Current log detail and accessibility. Future: Number of unexplainable errors, time to diagnose issues.
5. **Gaps in Non-Functional Requirement (NFR) Monitoring:**
    - **Finding:** A critical NFR (NF01) "Availability" are "Undefined" indicating a lack of current mechanisms or metrics to track uptime. Similarly, NF07 Maintenance with a Undefined status suggests that while development standards were aimed for, formal tracking or validation of code maintainability might be an area for improvement.
    - **Metric (To be established):** mean time between failures (MTBF), mean time to recovery (MTTR) or code coverage.

Implications for Future Developments (Learn Phase):

- **Prioritize Full Implementation of Core Features**: Focus on completing "Partially Implemented" functional requirements (FR02, FR05, FR09) and high-priority user stories to deliver a more rounded core experience.
- **Address User Feedback Systematically**: The "numerous suggestions" from officials should be cataloged, prioritized and integrated into the backlog.
- **Develop Comprehensive Logging**: Improve the logging system for better audit trails, easier debugging, and accountability, fulfilling FR09.