Product Installation and Maintenance Packages

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

June 20th, 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. Introduction

This document provides all the necessary information for the installation, configuration, and maintenance of the ZScore platform. The process is streamlined and relies on Docker and Docker Compose to manage the application's environment, ensuring consistency and ease of setup. The following sections detail the prerequisites and the specific commands required to build, run, and access the different components of the system, including the main application and its documentation.

2. ZScore Overview

ZScore is a platform for indoor scoreboards management and display. It provides built-in visualization and management tools for Basketball, Volleyball, and Futsal scoreboards.

3. Source Code Repository

The complete source code for the ZScore platform is hosted in a Git repository. The repository can be accessed at the following URL:

- **Repository Link:** https://github.com/FEUP-LGP-2025/LGP-17

4. Installation

4.1. Prerequisites

Before proceeding with the installation, ensure the following software is installed on your system:

- **Docker:** https://www.docker.com
- **Docker Compose:** https://www.docker.com/compose/

4.2. Installing Docker

The recommended method for installing Docker is to follow the official guide for your operating system. For Linux distributions like Ubuntu, you can find the guide here: https://docs.docker.com/install/linux/docker-ce/ubuntu/

Please follow the steps in the **"Install using the repository"** section of the guide.

After installation, it is highly recommended to complete the post-installation steps to manage Docker as a non-root user, which improves security and usability. These steps can be found here: https://docs.docker.com/install/linux/linux-postinstall/

4.3. Installing Docker Compose

To install Docker Compose, please follow the official instructions available at the Docker documentation website: https://docs.docker.com/compose/install/

5. Usage and Maintenance

5.1. Running the Application (Development Environment)

To start the application, you first need to build the Docker images. Navigate to the root of the project repository and run the following command:

```
docker compose build
```

If you have previously built the application and need to perform a clean rebuild, you must first remove the existing Docker volumes to avoid conflicts. You can do this with the following command:

```
docker compose down -v
```

Once the initial build is complete (or if you are performing a clean build), you can start the entire application stack using:

```
docker compose up --build
```

This command will start all services defined in the docker-compose.yml file.

5.2. Accessing the API Documentation

The project includes comprehensive API documentation. To view it, navigate to the docs folder located inside the be/api directory and run the following commands to install dependencies and start the documentation server:

```
npm install
npx docusaurus start
```

After the server starts, you can access the documentation by opening **http://localhost:3000** in your web browser.

6. Project Structure

The project is divided into two main components: the front-end and the back-end. Each can be explored and worked on individually.

- **Front End:** Detailed information can be found in fe/README.md.
- **Back End:** Detailed information can be found in be/README.md.

The repository has the following high-level structure:

```
.
├── fe/      # Contains the web application source code.
└── be/      # Contains the back-end services.
    ├── api/   # The API for processing information.
    └── db/    # Database configuration and data.
```