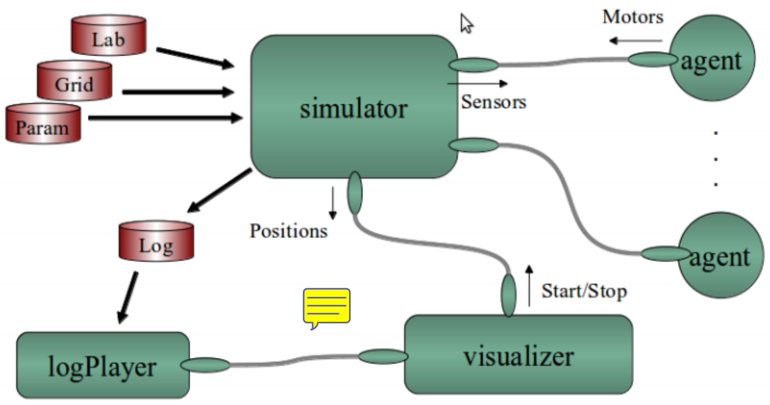
Deliverables

Authors: António Rafael Ferreira (67405), Bernardo Silva (67413), Bruno Silva (68535), Diogo Silva (60337), Pedro Gabriel Silva (68021) and Rodrigo Cunha (67800)
Abstract: This document contain a merge of all deliverables (project scope, main features, success criteria, use cases, deployment diagram, issues and risks, component diagram, system-wide requirements, business process and target scenarios). Basically most of the needs to describe our project.

Project Scope

Overview and the current state of the simulation tools CiberRato



The DETI/IEETA/IRIS Lab has a working group since 2000 that develops, maintains and uses the simulation environment of robotic agents CiberRato.
This environment has been used as a platform to support the various competitions in robotics, simulated, and in the support of the teaching of different courses, in both cases inside and outside of the UA.
The simulation environment allows you to create a virtual scenario where multiple virtual robots move to achieve a given goal.
The users of the environment are responsible for the development of the software that controls the virtual robots.

The software is structured in a distributed form, there are four main applications (simulator, viewer, agent, and logPlayer), which communicate (UDP/IP) by exchanging XML messages.
The main piece is the simulator, which is responsible for creating and managing all of the simulated world. The viewer is responsible for show graphically the state of the simulation.
The simulator can be instructed to record the messages exchanged with the viewer and the agents, and that data be used later to reproduce a simulation again, using the logPlayer.

Project Description

The aim is to enrich the CiberRato simulation environment in order to be effective and appellative. The desired remote interaction is developed in two levels: support for remote participation in CiberRato contests and creating a continuous competition environment.

The first goal it is intended that the remote participation of teams in CiberRato competition is possible by pr[illegible]ing appropriate means for submission of agents and the real-time monitoring of the simulations.
The second objective is intended to create a continuous competitive environment in which the participants can submit code to be executed periodically. The participant is informed of the score achieved and the best will be on a merit table (hall of fame).

Achieving these objectives requires the definition and implementation of a Web environment, with work at the server level, the customer level and depending on the progress of work at the level of the simulator.

Main Features and Success Criteria

NOTE: All statements are identified with **[C]** or **[T]**. **C** stands for complete, which means that the task is complete and already wor[illegible]g, **T** stands for 'To do' which means that the task is in progress or to do.

- ➢ Register the users, the platform allows the register of users wherein saves the submissions done previously by as well as respective scores. **[C]**
  - User should be able to change personal information (including password). **[C]**

- ➢ Competition Teams, is possible the creation of a team to the competition wherein all team members can see the code that was submitted and also submit, as well as the respective scores obtained, the team administrator still can manage the members that are in the team. **[C]**
  - Any user in a team should be able to associate an agent to that team. **[C]**

- ➢ There is no need for to the simulation of the agent (cibertools):
  - User is able to submit code for the simulator. **[C]**
  - User should be able to submit 3 different files:
    - Preparation script (That allows the server to prepare the agent, compilation mostly) **[T]**
    - Execution script (Indicates the server how to execute the agent and should have 3 parameters, first one is host, second one is grid position and last one is robot name. **[T]**

        - Source code (Agent) **[C]**
    - Use should be able to see the result of the simulation. **[C]**

- Code submitted by the user should be executed in a safe environment to prevent malicious user source code on the server. **[C]**

- Code submitted should be tested before accepted:
    - Arguments must be accepted (host, position and robot name) and valid. **[C]**
    - Preparation script should compile the code with success (if needed). **[T]**
    - Agent communicates correctly with the simulator (proper messages, proper port). **[C]**
    - Execution script should automatically initiate the robot inside the lab. **[T]**

- Besides the normal execution of an agent on the server-side, should be possible to execute the agent from the user side **[T]**. It means that the server should wait a short time allowing the user to execute the code on his machine:
    - The server should indicate the user the ip address where he must connect (where simulator is located). **[T]**

- Support on the competition, wherein in the day of the competition, all users registered will have the opportunity to submit their code to be reproduced at the right time by monitor responsible by the competition.
    - Different types of competition, competitive and collaborative:
        - In competition competitive are introduced robots in labyrinth wherein all members compete with each other to arrive to the goal in first. **[C]**
        - In competition collaborative are introduced robots in the same labyrinth wherein exchange information with each other to arrive to the goal more rapidly. **[T]**
    - Visualization of the competition, will be possible to some user that visit the platform visualize the competition that is taking place. **[T]**
    - The implementation of those types of competitions should be generic allowing to introduce new competitions easily. **[T]**
    - The user should be able to check all competitions that are going to happen or already happened. **[C]**

- Besides the visualization of the stream, should be possible to re-watch every submission that has been done, the log player. Log player should have the following characteristics:
    - Allow to set different speeds to the simulation. **[C]**
    - Allow the user to move forward or backwards (it implies that log player has a timeline). **[C]**
    - The user should be able to watch the sensors values. **[C]**

    - The user should be able to check information about all agents (score, sensors, positions). **[C]**

- Platform should have different roles with different permissions:
    - Normal user, which can manage every data related with himself and associated teams. **[C]**
    - Staff member, which can manage everything related with competitions. **[T]**
    - Administrator, which can manage everything related with competitions and manage user data (teams, agents, etc). **[C]**

- Simulator (cibertools) should have implement new characteristics to work better with the platform:
    - Output of the logger file and the messages sent to the viewer should have the same format since they're sending the same information. **[C]**
    - Three different entities to manage the simulator:
        - Viewer, which receives all the information about the simulation (robots location, scores, lab, grid, etc) **[C]**
        - Panel, doesn't receive any information about the simulation but he is able to reset simulation, start simulation, remove robots, send labs, send grids, send parameters, etc. **[C]**
        - PanelViewer, this entity is able to receive information about the simulation and is also able to send commands to the simulator. **[C]**
    - Simulator should work in daemon mode: **[T]**
        - Agents should have a time associated to themselves and not to the simulation. **[T]**
        - Log file should be associated to the agent and not to the simulation. **[T]**
        - Agents should have an communication channel associated to the robot and not to all the simulation. **[T]**

Ideas

- Hall of fame, platform should be able to record all the simulations and results. Meanwhile the top results will stay on a table, the hall of fame. This will allow the users to know the best scores ever done on platform.

- Online editor, the platform should have an editor online where the user is able to select the language he wants to program the agent. He should also be able to check the available API to program the agent.

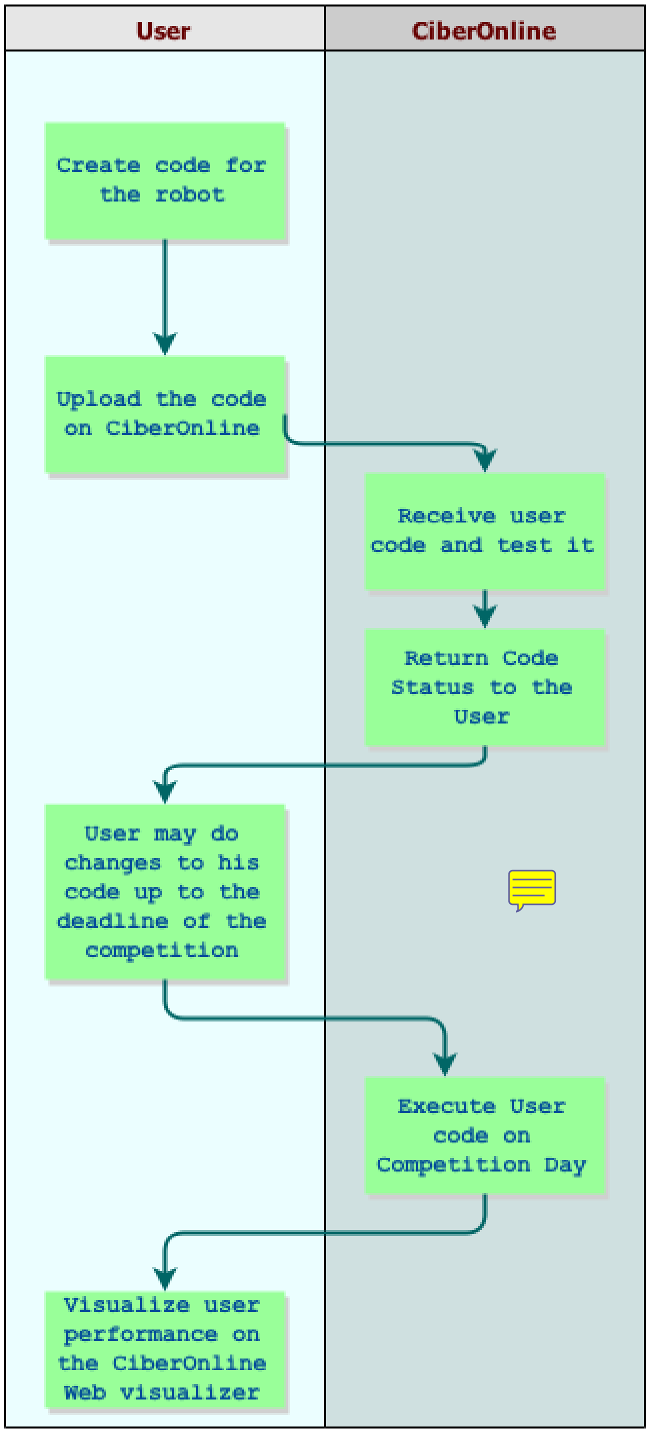
Issues and Risks

- Server failure due to an outage of a service (power, internet, ...)
- Packets not arriving on the client side and vice-versa due to internet traffic overload
- Malicious user code running on our servers
- Unexpected amount of user traffic
- Compatibility issues on the client browser choice

System-wide Requirements Non-Functional

- User needs a browser and an Internet Connection in order to use the service
- The service can be used on any browser with JavaScript support(Safari, Google Chrome, Mozilla Firefox, etc) on any OS
- After user's registration, he can, immediately, use his personal page without any difficulty
- The system should work continuously without delays
- The system should be easy to keep up to date
- The system will store the information of each user safely
- Teams can be created, where only the admin can make changes to it (add/remove user, etc)
- The system will be available in English
- System should support the upload of code in different languages
- The code should be executed on a controlled environment
- User should be able to use the system without any difficulty after his first experience

Business Process


User
CiberOnline
Create code for the robot
Upload the code on CiberOnline
Receive user code and test it
Return Code Status to the User
User may do changes to his code up to the deadline of the competition
Execute User code on Competition Day
Visualize user performance on the CiberOnline Web visualizer

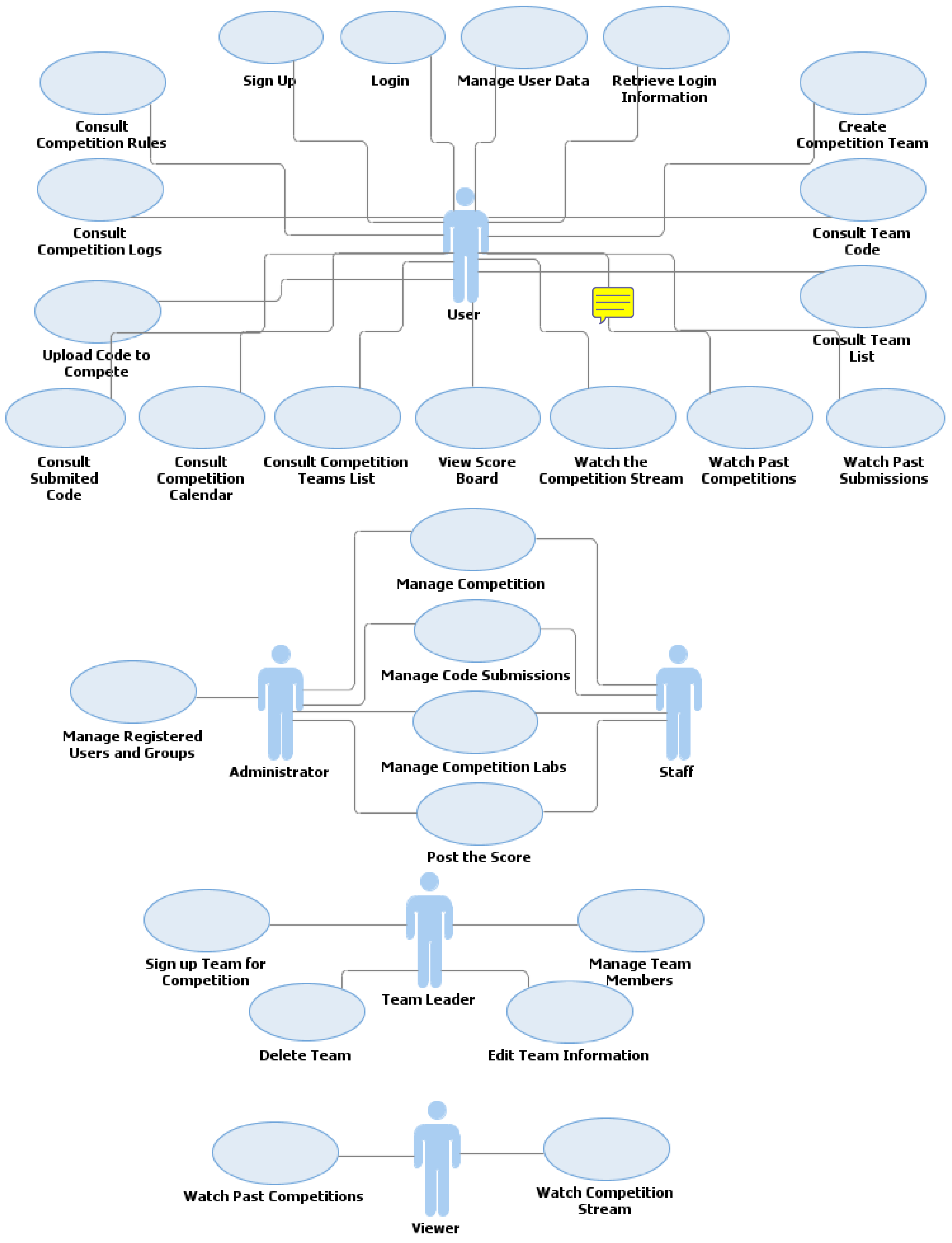
As we look at our business process we can see the user starts by creating their own code. After that, in order to test his code, he must upload the code to our system (CiberOnline) where it will be tested.

During the process of testing the user code, he will be notified if his code passed our test and is able to run on the competition day. After the user receive a confirmation that his code is good to run, he may still submit a new version of it up to the deadline established by the competition.

On the competition day the code is executed along with all other participants and the user may watch his performance on the CiberOnline Web Visualizer.

Use Cases

Use Cases Diagram



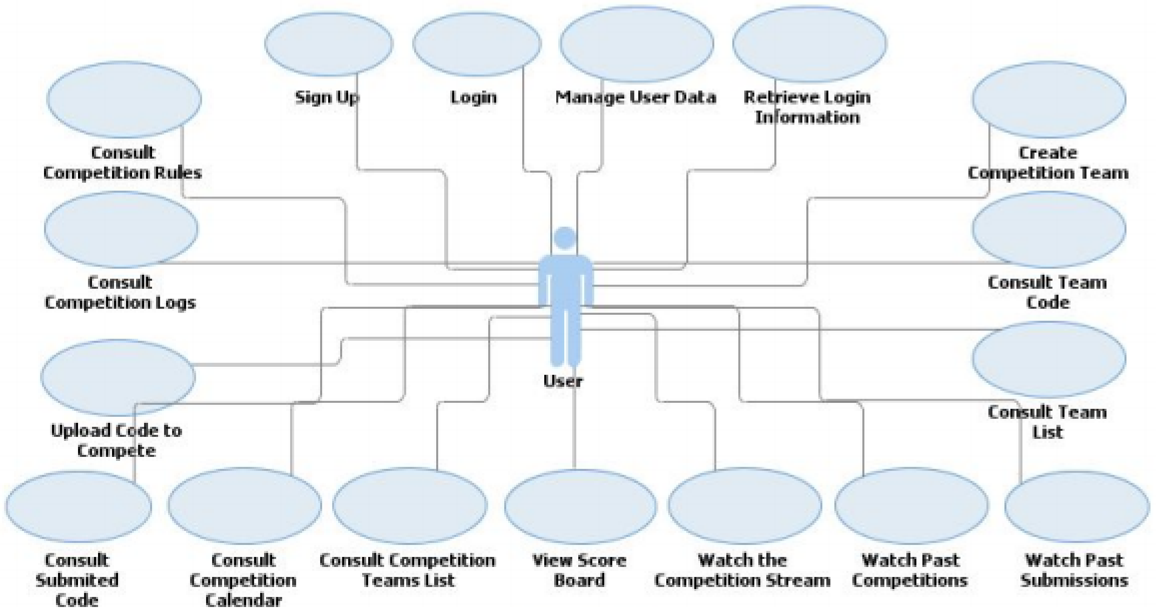
Use Cases

- **Login**
  - The user provides the email or username and password to login.
- **Sign up**
  - The user creates the profile and provides the email, username and password.
- **Manage User Data**
  - The user can edit his profile information (email, password, ect.).
- **Retrieve Login Information**
  - The user can request a new password that is sent to his email, if he forgets it.
- **Upload Code to Compete**
  - The user can upload his code to be used in competition against other users or teams.
- **Watch the Competition Stream**
  - The user can watch the competition live stream in the web viewer page.
- **Consult Competition Rules**
  - The user can consult the competition rules and FAQ page.
- **Create Competition Team**
  - The user can create a new Team that can compete against other users or other teams.
- **Consult Team List**
  - The user can browse the list of subscribed teams and consult information about them.
- **Consult Team Code**
  - The user can consult the code submitted by his team.
- **Consult Competition Logs**
  - The user can consult or download the competition logs.
- **Consult Submitted Code**
  - The user can consult or edit the submitted code.
- **Consult Competition Calendar**
  - The user can view informations about the competition dates and schedule.
- **Consult Competition Teams List**
  - The user can consult the list of users and teams that are in the competition.

- **Watch Past Competitions**
    - The user can browse the past competitions and can watch them in the web viewer page.
- **Watch Past Submissions**
    - The user can browse the past submissions and can watch them in the web viewer page.
- **View Score Board**
    - The user can view the score board of the current competition and the online competition.
- **Manage Team Members**
    - The team leader can browse, add or delete team members.
- **Delete Team**
    - The team leader can delete the team.
- **Edit Team Information**
    - The team leader can edit informations about the team(name, description, ect.).
- **Sign up Team for Competition**
    - The team leader can submit the code for the group participation in the competition.
- **Manage the Competition**
    - The moderator or the administrator can trigger(start) the competition, chose the labs and post the score.
- **Manage Code submissions**
    - The moderator or the administrator can browse and accept users or groups code submissions for the competition.
- **Manage Registered Users and Groups**
    - The administrator can browse and view the members list information, edit groups members, as well as delete users or groups.

Target Scenarios

User:



Login - In order to have access to the CiberRato`s page, if the user already have account, he must provide the email and the password to login. If the user forgets his password (**Retrieve Login Information**), he can obtain a new password that will be sent to he`s email.

Sign up - If user doesn`t have account and want to access to the page, he must create an account, providing first and last name, email, password and the educational institution. After that, the system will register his dates and then, they will be used in the login to have access.

Manager User Data - After the user have access to CiberRato`s page, if he wants to edit his profile information, he can do it using this option. He can edit your first and last name, email, password and educational institution. User can do this anytime.

Retrieve login Information - As was explained in the **Login`s** use case, the user can request a new password that is sent to his email, if he forgets it. Starting there, he must use this password to login to CiberRato`s page. There`s no password change requests limit.

Create Competition Team - In user`s page, he can create a new Team that can compete against other users or other teams. For this, user must provide team`s name and number of elements to the team. Lately, the Team Manager can add other members to the team.

Consult Team Code - Here, the user can consult the code submitted by his team. For the teams that will participate in some competition, all teams must submit his code and lately can consult it.

Consult Team List - The user can browse the list of subscribed teams and consult information about them like the name and the number of elements of these teams. User can do it anytime.

Watch Past Submissions - The user have access to the web viewer page and here, he can browse all past submissions of his team and can watch them.

Watch Past Competitions - After some competition is over, user have access to the log and here user can browse the past competitions and watch them in the web viewer page.

Watch the competition stream - In the competition day, upon submitted the code, the user can watch the competition live stream in the web viewer page.

View Score Board - The user can view the scoreboard of the current competition and the online competition. In any competition, users have a score that is registered.

Consult Competition Teams List - The user can consult the list of users and teams that are in the competition. He can consult all teams that participated in last competitions and in the current competition.

Consult Competition Calendar - The user can view informations about the competition dates and schedule.

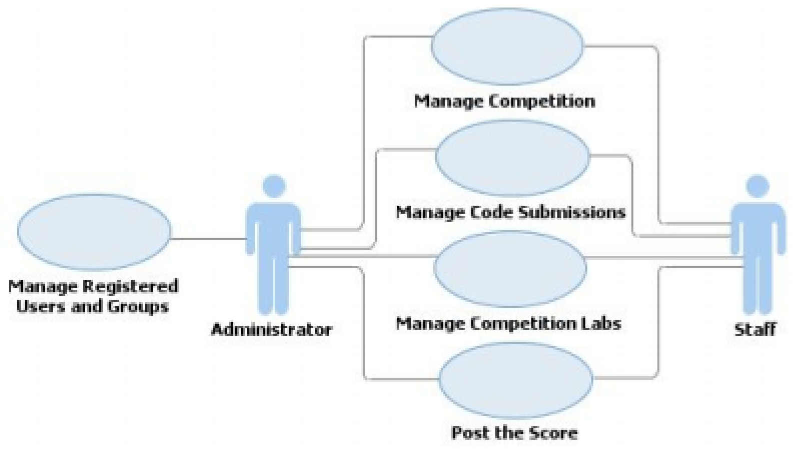
Consult Submitted Code - The user can consult or edit the submitted code for the competition. For the last submissions in last competitions user only can consult them.

Upload code to Compete - In order to compete, the user can upload his code to be used in competition against other users or teams.

Consult Competition logs - After the competition is over, immediately, the user can consult the log competition or download the competition logs.

Consult Competition Rules - In order to keep the user informed, he can consult the competition rules. If the user doesn`t obey some rule, the Admin can banish him.

Admin and Staff:



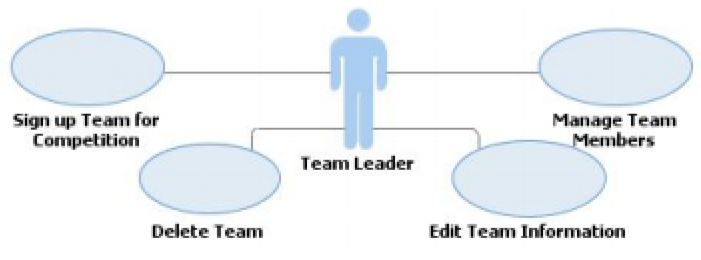
Manage Registered Users and Groups - The administrator have full control of the page and he can browse and view the members list information, edit groups members, as well as delete users or teams. He can also create a new team add members for the created team.

Manage the Competition - The moderator or the administrator can trigger(start) the competition, chose the labs and post the score. He can also add teams or users to the competition or delete them from the competition.

Manage Code Submissions - Before the competition begins, the users or teams must submit its code that will be used by its agent in the competition and the moderator or the administrator can browse and accept users or groups code submissions for the competition.

Post the score - After the competition is over, the administrator will post the score in the Scoreboard and update it for later user can consult it. The score is evaluated by the number of collisions and the starting time that the agent reaches the beacon to his starting position.

Team Leader:



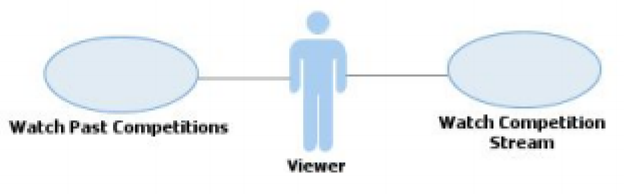
Sign up Team for Competition - Before the competition begins, the team leader can submit the code for the team participation in the competition.

Manage Team Members - The team leader can browse, add or delete team members but, although he can manage team members, he can`t edit members profile! This duty is only done by the administrator.

Delete Team - The team leader can see the team information and delete them or delete some members of the team.

Edit Team Information - The team leader can edit informations about the team(name, description, ect.). This use case is included in Manage Team Members`s use case.

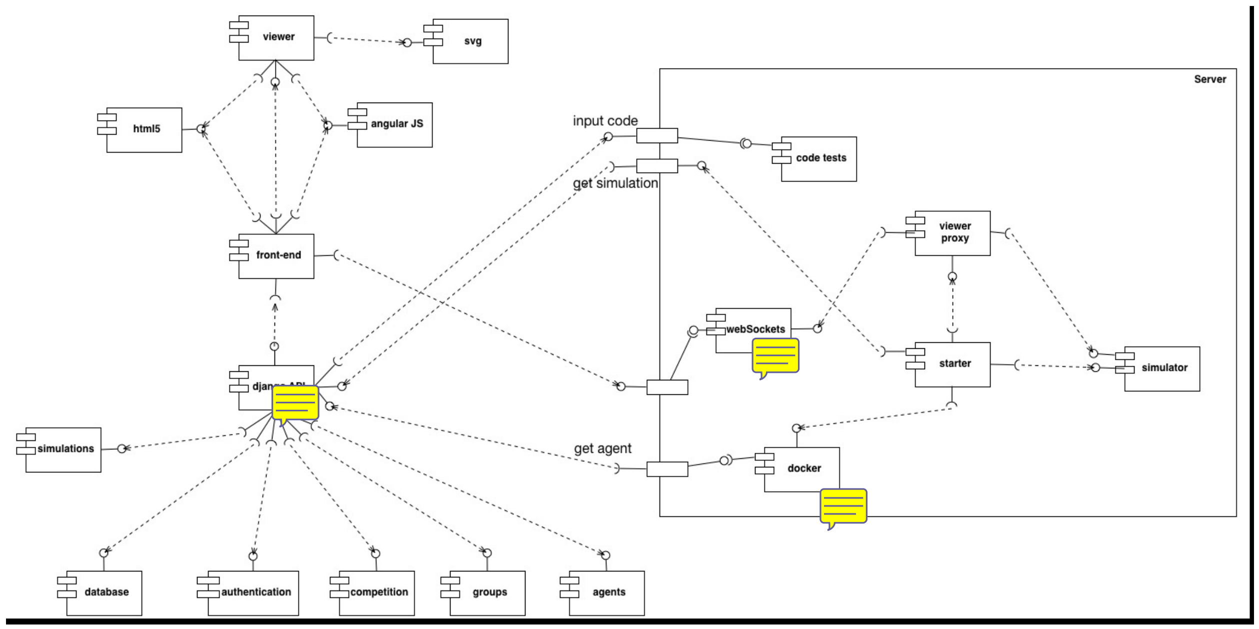
Viewer:



Watch Past Competitions - After some competition is over, the viewer have access to the log and here her can browse the past competitions and watch them in his web viewer page.

Watch Competition Stream - In the competition day, upon user submitted the code, the viewer can watch the competition live stream in his web viewer page.

Component Diagram



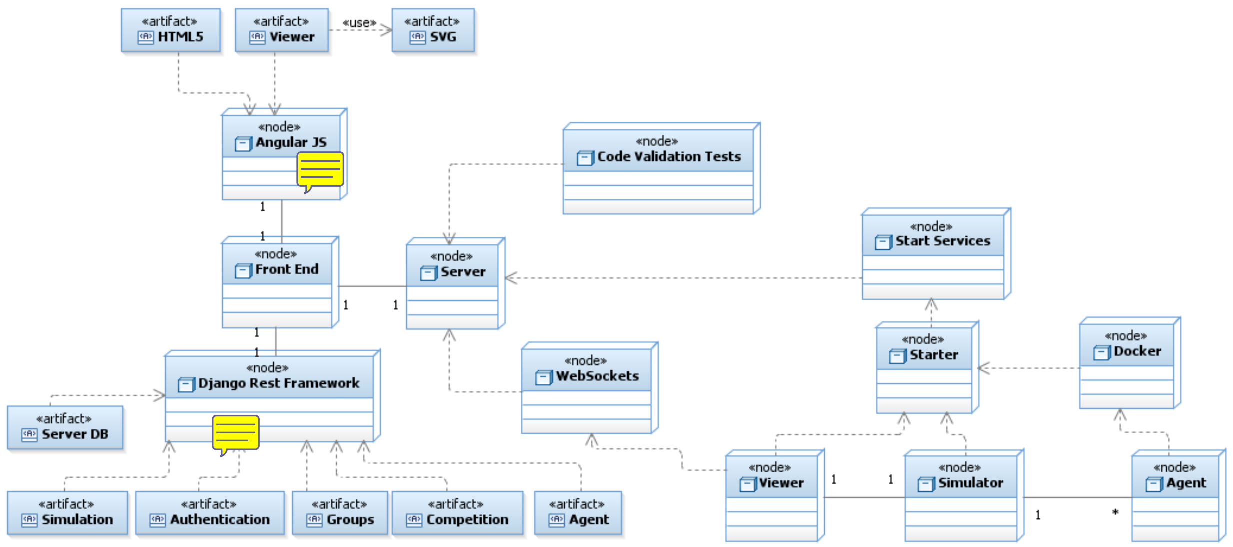
The component diagram described in the picture above represents the components present in the CiberRato online platform.

Starting from the upper left corner we have our web viewer that uses modules from both svg, html5 and angularJS. Below that it's our front-end with handles with the interaction with the users and also uses modules from html5 and angularJS.

When we move a bit more to the lower left part we have our API developed in Django and it handles all the simulations, databases, authentication, competitions, groups and agents providing end-points to all of those components. All the interaction with the server, which is on the right hand side, is done through the API.

Finally on the right side we have the server, which has smaller components inside. The code tests is responsible for verifying the correctness of the code and communicates with the API both for receiving the code as well as providing feedback from the tests. The websockets are responsible to send the data related to the simulation live and interact with the viewer that receives the simulation status from the simulator and sends it to the websockets as well as stores it. The starter is responsible to start the simulator, the viewer, the websockets and the agents. For this last one it executes them in a safe environment using the docker component. The last component is the simulator which is responsible for simulating the competition and send the results to the viewer.

System wide architecture - Deployment



In this diagram it's possible to see the components connected to the nodes of the system (UML Deployment Diagram).

There are some important path communications to consider. The server side has 7 nodes: Websockets wich is responsible to comunicate with the Front End to delivery the real log from the Viewer. The Start Services which is responsible to start the simulation and other tasks, the Viewer which is responsible to register at the Simulator and receive the data generated and to run the Agent inside a Docker environment.

The code validation tests is a node to validate the Agent code before be available to run in a simulation.

The communication from the server to the front end is made through the Django Rest Framework, only the WebSockets is directly from the Server side. The Django Rest Framework has those nodes: Server DB, Simulation, Authentication, Groups, Competition and Agent .The Server DB is responsible to store the data. The authentication app is responsible to authenticate the users and manage the user information. The groups app is reponsible to manage the competition groups. The competition app is responsible to manage all the actions possible to make in the competition. The Simulation app is resposible to handle all simulations and Agent app to handle all agents actions.

The Front End is handled by AngularJS and the Viewer uses SVG. This artifacts are resposible to retrieve the data from the Front-End and present them to the user.