Project Number: 101048546

Project Acronym: MPCS

Project title: Marine Pollution Control Simulator

Simulator Structure Design Report

Deliverable 3.1

Version 1.0

14 March 2023

Table of Contents

Index of tables

Table of figures

Acronyms and Abbreviations

AMN – Autoridade Marítima Nacional

AP – Administração Portuária

CCC – Centro de Comando e Controlo

COMAR – Centro de Operações Marítimas

DCPM – Direcção de Combate à Poluição do Mar

DGAM – Direcção Geral da Autoridade Marítima

EMSA – European Maritime Safety Agency

GNR – Guarda Nacional Republicana

HC – Hidrocarboneto

Mohid – Water Modelling and Simulation System

MPC – Marine Pollution Control

PM – Polícia Marítima

POLREP - Pollution Reporting System

PSP – Polícia de Segurança Pública

SAM – Sistema de Autoridade Marítima

History of Changes

Table 1: History of Changes

Version	Publication date	Changes
1.0	14.03.2023	First version

• Contractual aspects

Project name:	Marine Pollution Control Simulator (MPCS)
Project number:	101048546
Deliverable:	D3.1 – Simulator Structure Design Report
Work package:	WP3 – Design and Development
Task:	3.1 - Simulator structure design and development
Dissemination Level:	EU - Sensitive
Version:	1.0
Contractual Date of Delivery to the EC:	28.02.2023
Actual Date of Delivery to the EC:	17.03.2023
Leader entity:	UNIV. COIMBRA
Participant(s):	QUALISEG, IPTL, EVM, DGAM
Author(s):	Licínio Roque, Luís Lucas Pereira, Ana Sobral, Maria Dias, Miguel Lopes, José Sobrinho, João Barata, Jorge CS Cardoso, Fernando Barros

• Legal Disclaimer

The project Marine Pollution Control Simulator (MPCS), No. 101048546, has received funding under the Union Civil Protection Mechanism, Call: UCPM-2021-PP — Prevention and Preparedness Projects on Civil Protection and Marine Pollution, from the European Union (EU), represented by the European Commission (EC).

1. Introduction

This report presents the preliminary Software System Architecture specification for the Marine Pollution Control Simulator (MPCS). This first model draft will work as a living document to be improved during the software development process.

Software system architecture focus on the definition of components and interconnections for the system composition under regular operation. It includes specification of library modules and subsystems required to operate the software developed such as database management systems and their data models. Therefore, they are significantly influenced by design decisions, requiring adjustment as new system considerations are identified during development.

Chapter 2 summarises the MPCS concept. Subsequently Chapter 3 describes the initial proposal for MPCS Software System Architecture, including its modular components and interactions. Chapter 4 will present the required Data Model. Chapter 5 will describe the data entities and their intended use. Chapter 6 discusses the development environment. Chapter 7 briefly introduces applicable quality attributes.

2. The MPCS Concept

The MPCS should be a multilingual environment (Portuguese, Spanish and English), based on a Platform as a Service, for teaching, training and performance evaluation, individually and in teams, of actions to combat maritime pollution.

Teaching should consist of a set of e-learning courses on Combating Maritime Pollution, on the operation of the Simulator and on the management of the Simulator. The performance evaluation should allow the evaluation of the users' performance in the training.

The training (simulation) will take place in a multi-user environment in the Cloud, where each user will assume the role of a real professional belonging to the real existing device made up of organisms, facilities, people and equipment. The simulation will run on a given real geography, based on a Maps platform and on real weather conditions recorded from the relatively recent past. Users will be able, and should, interact with each other, interact with the equipment and interact with the spill in order to achieve the objective of the Exercise. A mathematical model of the spill and its interaction with the elements and combat actions will simulate reality enough and enough for the MPCS objectives.

The MPCS should provide 2 types of Exercise, Exercise for teaching and Exercise for training. The first is aimed at schools and professionals in training. The second is intended for professionals to maintain high readiness.

Users should be able to run the Exercise via their smartphone, tablet or computer.

In the case of Workout for Workout, your login will determine who you are in Workout. Each available profile will, in fact, correspond to one of the elements of the existing device, as well as the bodies, installations and equipment. The responsibility for keeping these data updated will be the responsibility of the **MPCS Manager** and the adhering entities.

In the case of the Exercise for teaching, after logging in, the user will be able to choose, among the available virtual people, which person is in the Exercise. The bodies, facilities, people and equipment may be fictitious and be created and configured for each Exercise by the MPCS Manager.

Users, when assuming an identity, will inherit the properties that the database has defined for that identity, which will differ from person to person (eg qualifications, roles, contacts, Hxh cost, ...).

All users, in the role of a certain virtual person, will be able to move virtually, and thus change their geographic location, will be able to rest and eat to recover their virtual health (property that varies with the hours of work, rest, eating or random injury), and will be able to virtually interact with other people and equipment. For example, communicate with each other, in an auditable manner, by typified and protocol means (voice, mobile phone, VHF, UHF, SIRESP, Satellite Communication or computer over the Internet) in a simulated way.

Any interaction between the different virtual entities must be determined by some of their current properties. For example, if a user wants to travel from the Lisbon Naval Base to Cascais, to board a speedboat, he will have to approach a vehicle and, in order to drive it, he will have to have health above a certain value and qualification to drive (license driving this type of vehicle) otherwise, you can only be a passenger waiting for a person with a license.

The virtual entities – person and equipment, must interact and move in real geography and infrastructure (maps platform) and real (recorded) meteorology. For example, the vehicle in the previous example must be moved at a speed, normal or urgent, defined for that type of vehicle, by the fastest road suggested by a maps platform or by the route defined by the user. The virtual location of the vehicle, driver, its passengers and transported cargo will be updated in the respective properties (location) at each moment of the simulation, taking into account the road, where they are moving, and speed.

The available interactions should allow the simulation of the necessary and fundamental procedures in the fight against pollution, such as the installation of equipment (barriers, recuperators, pumps, shovels, hoses, blankets, …), communication between people, the movement of people and equipment, refuelling, maintenance, etc.

The spillage drift, the hypothetical effect on the coast and the effect of equipment on its drift, as well as its quantity, should be based on a realistic mathematical model and on actual geography, meteorology, and hydrodynamics, as well as actual material specifications.

3. MPCS Software System Architecture

3.1. MPCS Architecture (Context View)

This section presents the first version of MPCS software system architecture, covering aspects of the platform's component architecture, intended users and access mode, data model for storage, among other aspects that configure the software development process.

The MPCS System is cloud-based and provides web-based User Interfaces (UIs) that users can access through any browser-based system (Fig. 1).

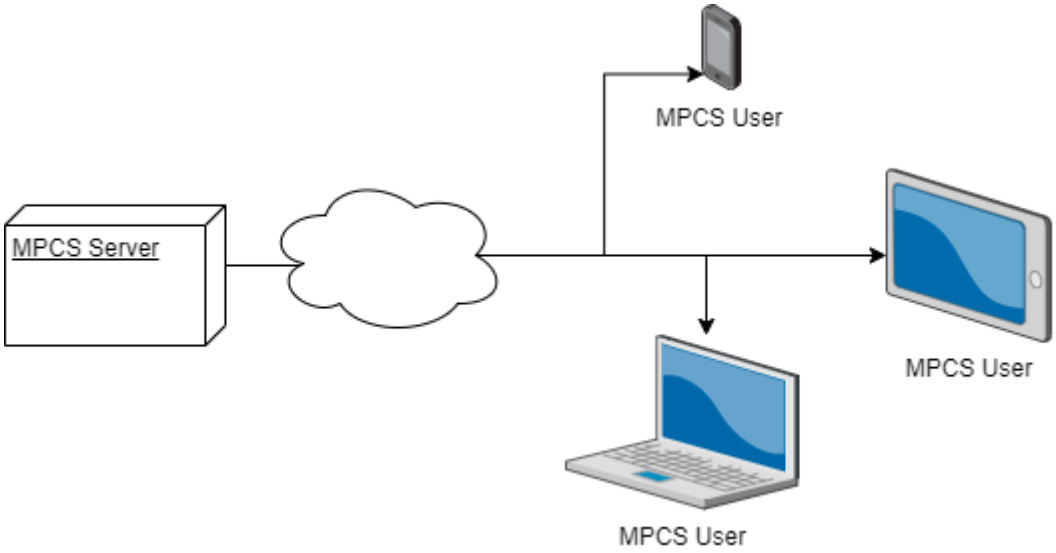

Figure 1: Overall System Architecture

3.2. MPCS Software System Components

MPCS Software System Architecture follows a multi-tier client-server model, composed of

- MPCS Game Editor - enables the configuration, management, and data exploration
- MPCS Game Framework - enables the UI components to act and update game state
- MPCS UI Generator – generates appropriate UI interfaces for each simulation role
- MPCS Digital Twin – integrates MOHID, manages and updates simulation state
- MPCS VR Interface – exploratory UI component for operational training (one of UIs)

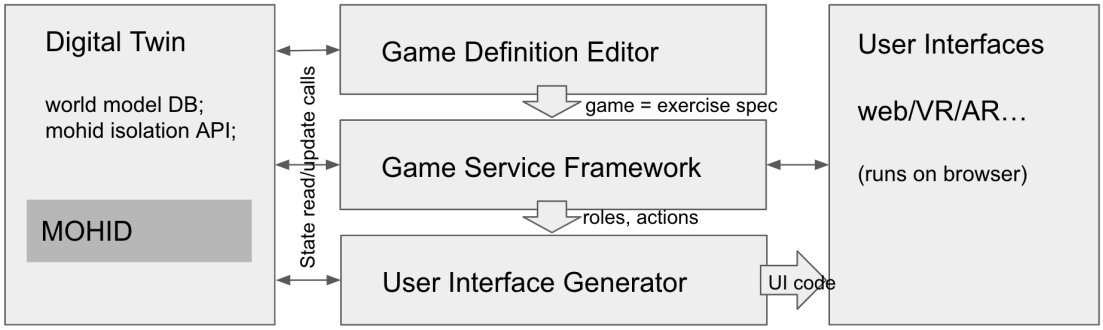

Figure 2: MPCS Architecture Components

The rationale behind the multi-tier client server architecture is to enable separation of concerns between definitions of one core game simulation infrastructure accessible able to enable access from multiple Internet and web browser enabled equipment (desktop and mobile). A variety of interfaces must be served according to the target equipment and participant role and state in the exercise. New and exploratory participant interfaces can be developed to reuse the base simulation system.

In this way a central game state will be used as the basis to evaluate possible actions and their results, enabling validation and calculus of the coordination of roles along each simulation game exercise. A hydrodynamics drift simulation model will be encapsulated to provide the physical process simulation of hydrocarbons spill and drift, behind an extended game state definition that includes persons, resources and equipments available to the exercise at each time.

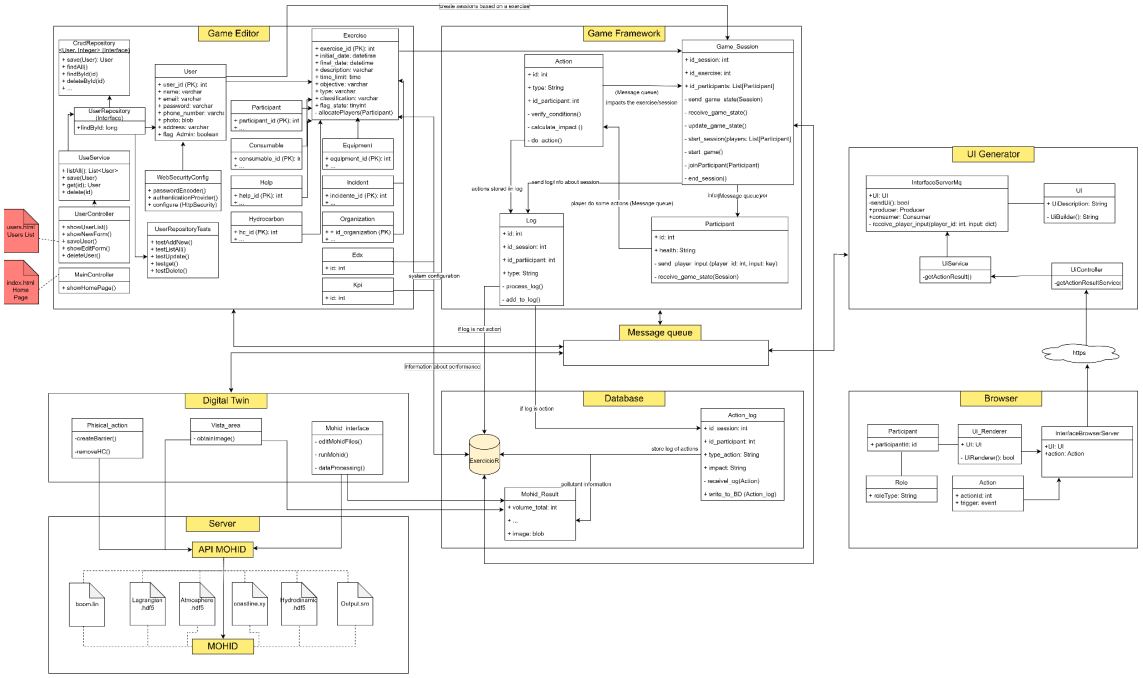

Figure 3: MPCS Software System Components

3.3. MPCS Game Editor Component

The MPCS (Exercise) Game Editor component enables:

- infrastructure operators to configure the operational conditions available;
- exercise managers to configure the conditions for each simulation exercise, to manage participants, to start and monitor the exercise flow;
- exercise managers to obtain performance reports on each exercise.

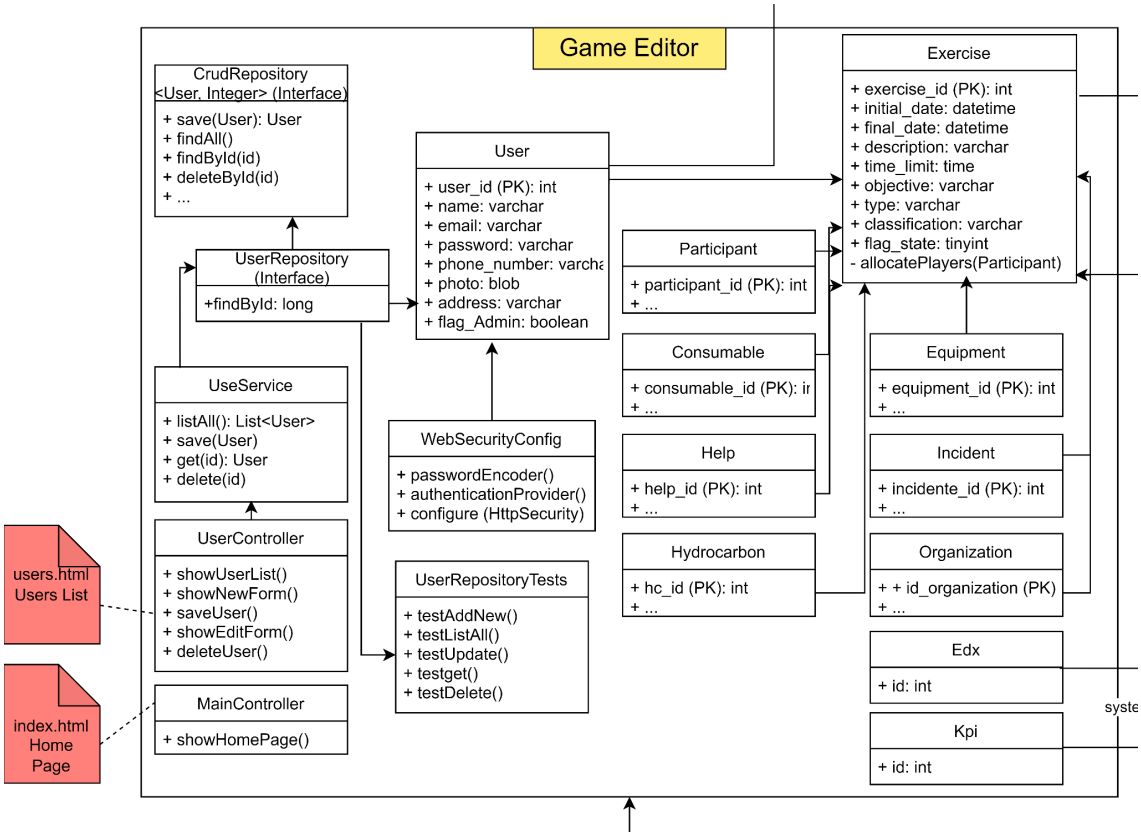

Figure 4: MPCS Game Configuration Editor Component

The Class Exercise is an important component of a larger system as it is connected to several other classes including User, Participant, Consumable, Help, Hydrocarbon, Equipment, Incident, and Organization. Each of these classes plays a crucial role in allowing the class Exercise to work properly. All of these classes are closely interconnected, and the class Exercise can only proceed when all of them are properly configured and integrated.

The UserController is responsible for handling HTTP requests related to the User class. UserService is responsible for implementing business logic for the User class. It provides a layer of abstraction between the UserController and the UserRepository. The UserRepository is responsible for interacting with the database for the User class. These three components work together to provide an abstraction layer for the User class, separating business logic from database interaction and user interface. Spring Boot provides built-in resources to facilitate

the configuration and development of these components, such as dependency injection, object mapping, exception handling, and more.

WebSecurityConfig is a class in the Spring Security framework that is used to configure security settings for a web application. KPI (Key Performance Indicator) is a performance metric that measures progress towards specific goals.

3.4. MPCS Digital Twin Component

The MPCS Digital Twin component's main responsibility is to emulate the interaction with the exercise physical conditions. This will be achieved by modelling relevant scenario variables and by integrating the MOHID (hydrodynamics drift model simulator) component.

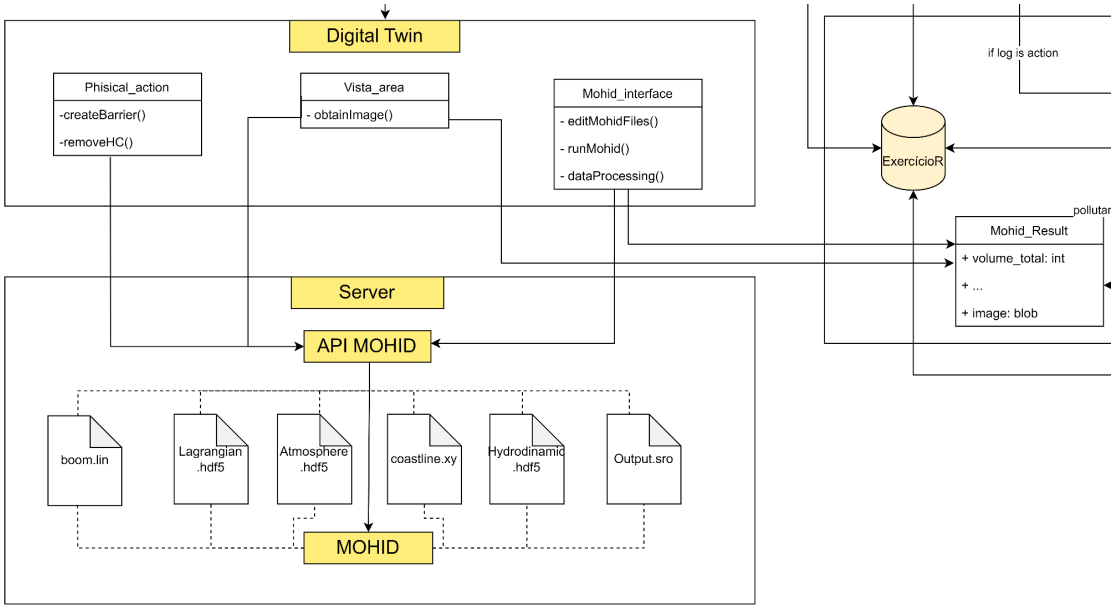

Figure 5: MPCS Digital Twin Component

The Digital Twin component provides methods for configuring the MOHID simulation parameters, such as specifying the start and end dates and times, setting up barriers for containing the spill, and indicating the spill location, the spilled hydrocarbon properties and equipment and personnel coordinates.

The MPCS Digital Twin component provides several API methods, such as placing barriers, removing hydrocarbons and editing configuration files, to allow for MOHID calculations. After this, the module processes the data and stores the results in the database.

During the exercise, this module is used for image acquisition in order to understand the evolution of the spill over time. It also recalculates the spill's state when users use equipment that changes the current state.

The following model provides a more detailed representation of MOHID interactions. MOHID uses data from terrain cartography, hydrodynamic conditions, weather conditions, and

pollution event data to calculate the spill event's evolution. The Mohid API utilizes the database to modify configuration files and to call MOHID. In the digital space, a Digital Twin Intelligent Model is also represented as part of another module of the solution that uses KPIs to evaluate performance.

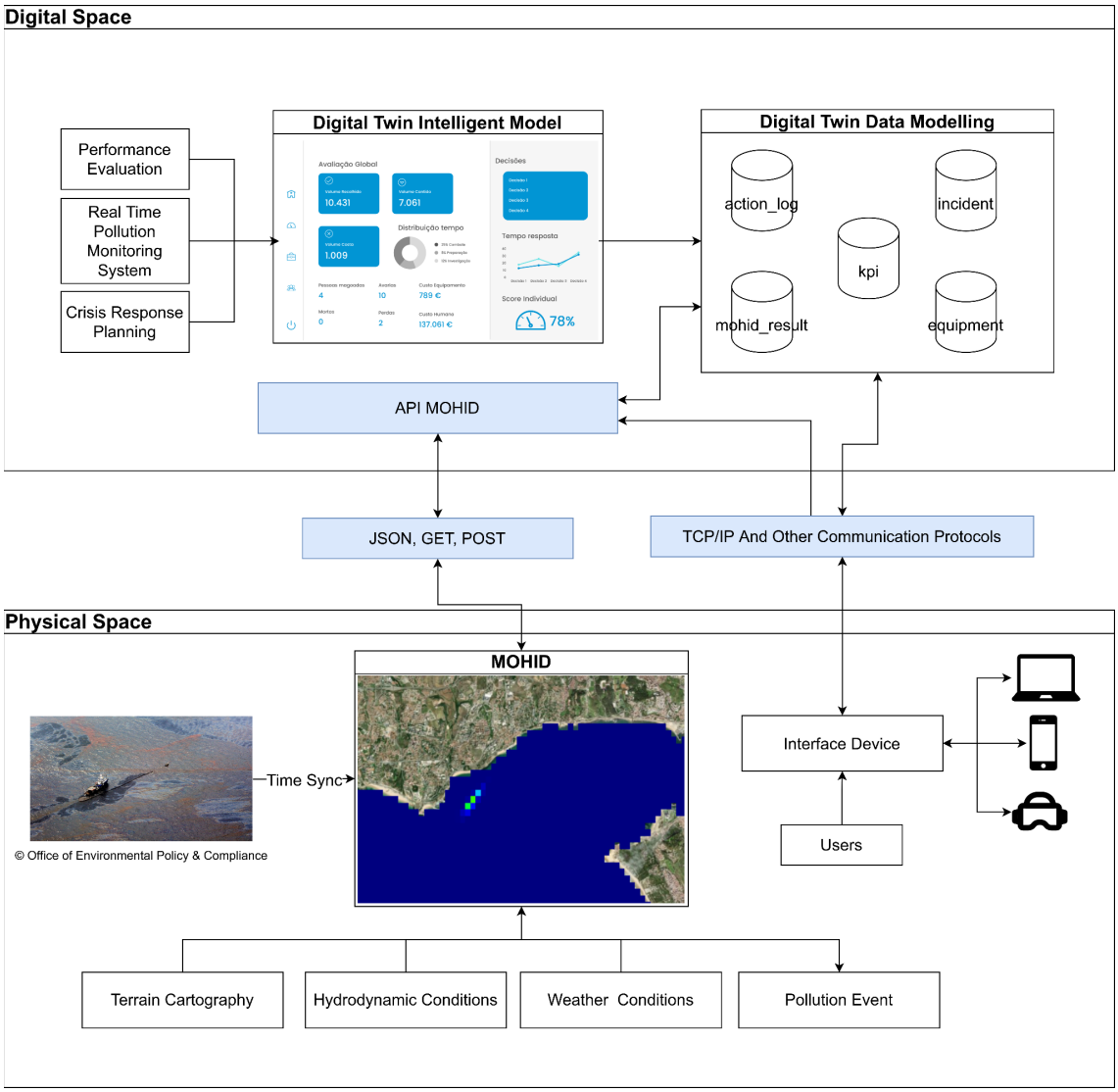

Figure 6: MPCS Digital Twin Model

3.5. MPCS Game Framework Component

The MPCS simulation platform runs each exercise based on the configuration and state data stored with the data model. The MPCS Game Framework responsibility is to welcome each participant, distribute the appropriate UI interface, accept event/action requests and process them accordingly, by coordination the other MPCS components. In the process the exercise (or game) state will be updated according with action pre-conditions and post-conditions.

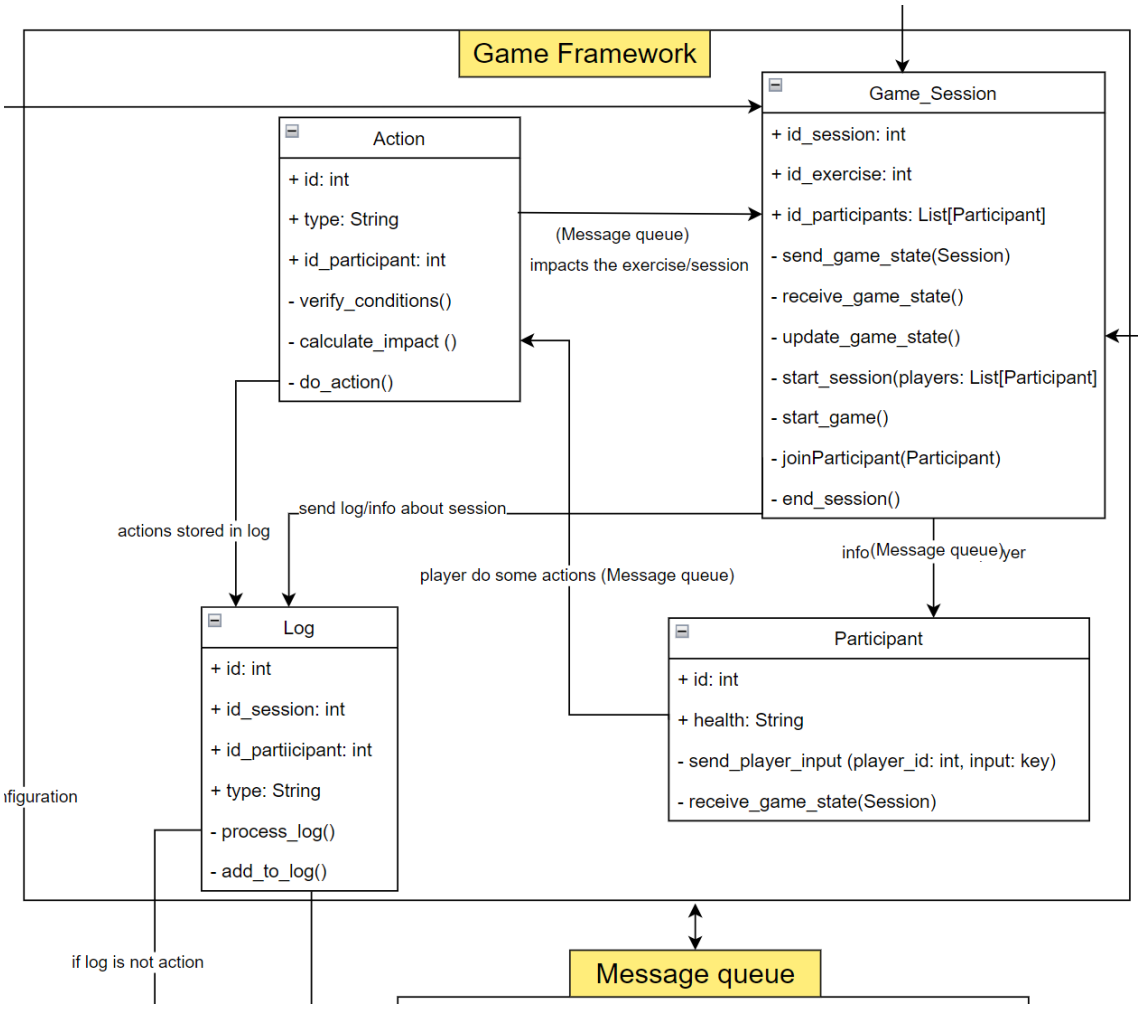

Figure 7: MPCS Game Service Framework Component

In order to create Game Sessions, the User (defined a priori in Game Editor module) will define which exercise (also already defined) will be incorporated in each determined session. After this the User (only the MPCS Manager can do this actions) will allocate players (a function in Exercise class) and Game Session will only start the game when already all participants have joined the session (they will be in a “lobby” waiting for all players).

In each Game Session, Participants will do actions (provided by the input in the Interface Module, via Message Queue) that have consequences, thus impacting the game world and state (i.e, the spill or equipments and players). This actions have conditions that need to be validated. Also, for each session, a log report is issued. This report can have periodically logs

provided by the system (i.e, the spill recovered, date, etc.) or actions log (when a certain player does an action this action and its possible impact is recorded).

This Game framework module interacts with a Database Module, which main objective is to store all data needed of the system and for each session (updating game world and game state).

3.6. MPCS User Interface Generation Component

The MPCS simulation platform enables a set of diverse roles to be performed in each training exercise. The MPCS UI Generation component has the responsibility to generate and manage the appropriate User Interface component for each participant according to the role and actions enabled by the exercise conditions.

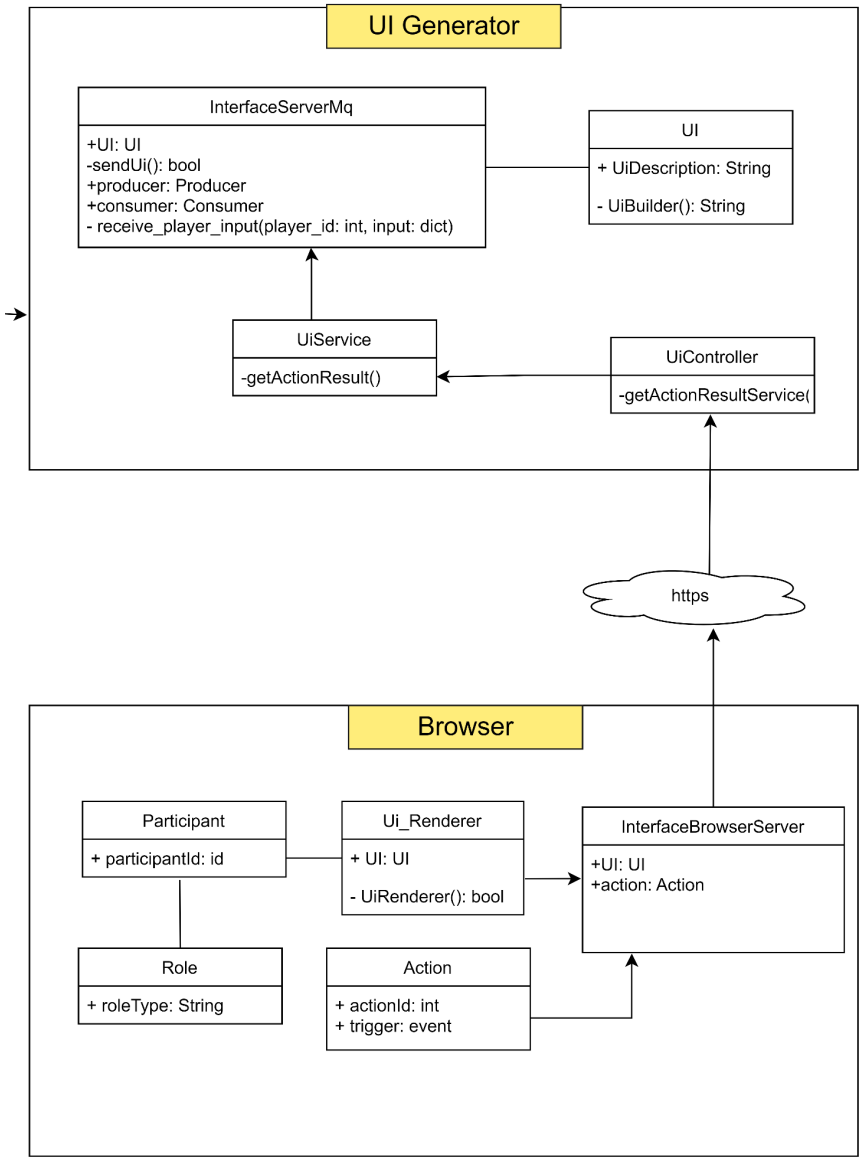

Figure 8: MPCS User Interface Generator Component

The UI Generator component contains a few classes such as the UIController and UIService, that are in charge to establish and communicate with the browser. Also, through the class InterfaceServerMq, this component communicates with the message queue in order obtain needed data to describe and generate an interface. With this received data, the class UI builds the User interface based on the data received, and finally tha class InterfaceServerMq sends the requested UI back to the browser to be rendered.

3.7. MPCS Component Interconnection

The MPCS components operate following a multi-tier client-server model were

- user actions are generated over a web browser based UI interface
- the UI components interact with Service Interface components managing each participant session.

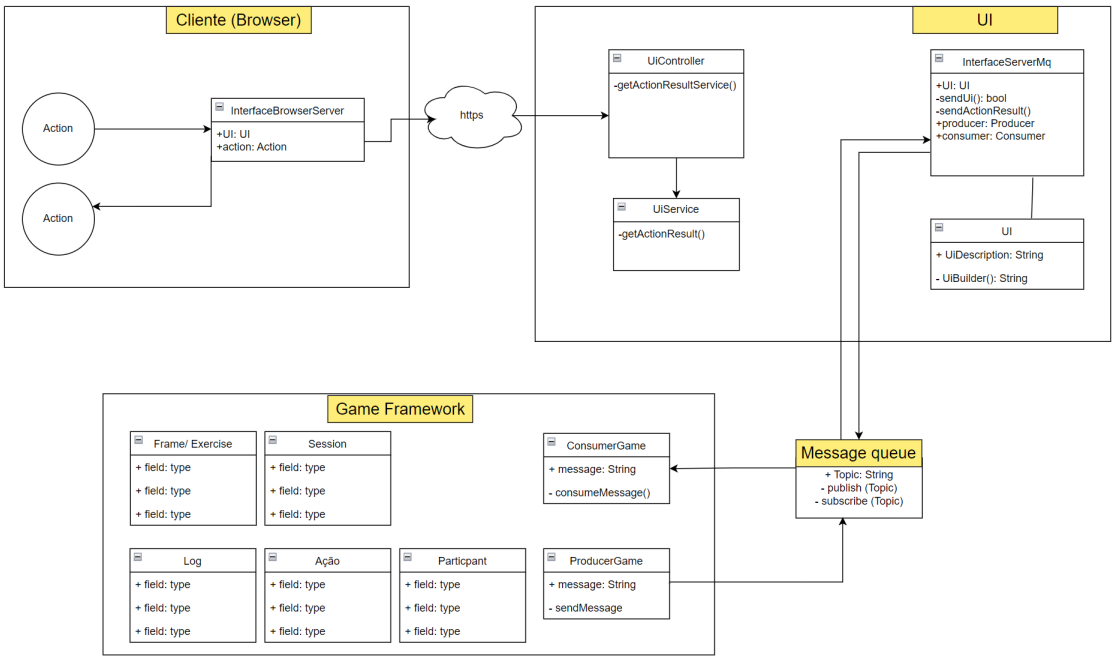

Figure 9: MPCS Components Interconnection

The UI Generator component communicates directly with the client (browser), by using Spring boot, in order to provide the most adequate graphical interface for the needs of the user. To accomplish this, whenever there is an action in the browser that needs a new/change on the UI, there is an https request to UIController that is in charge to map every endpoint for the respective services defined in UIService.

These services will invoke the adequate method defined in the class InterfaceServerMq. This class will send a message to the message queue so that it can receive the needed data in order to generate the appropriate UI. Upon the receiving of the data, this class processes the data

and generates a description of the interface and sends it back to the browser, where the UI description is rendered by the class UIRenderer(), and finally to the user.

3.8. MPCS Message Queue and Database Interface Components

The MPCS include a Message Queue, created using RabbitMQ, for the purpose of abling the communication between every component. We can accomplish this by defining topics where two or more components can communicate in order to exchange data and consequently alter the state of the simulation satisfying the user needs.

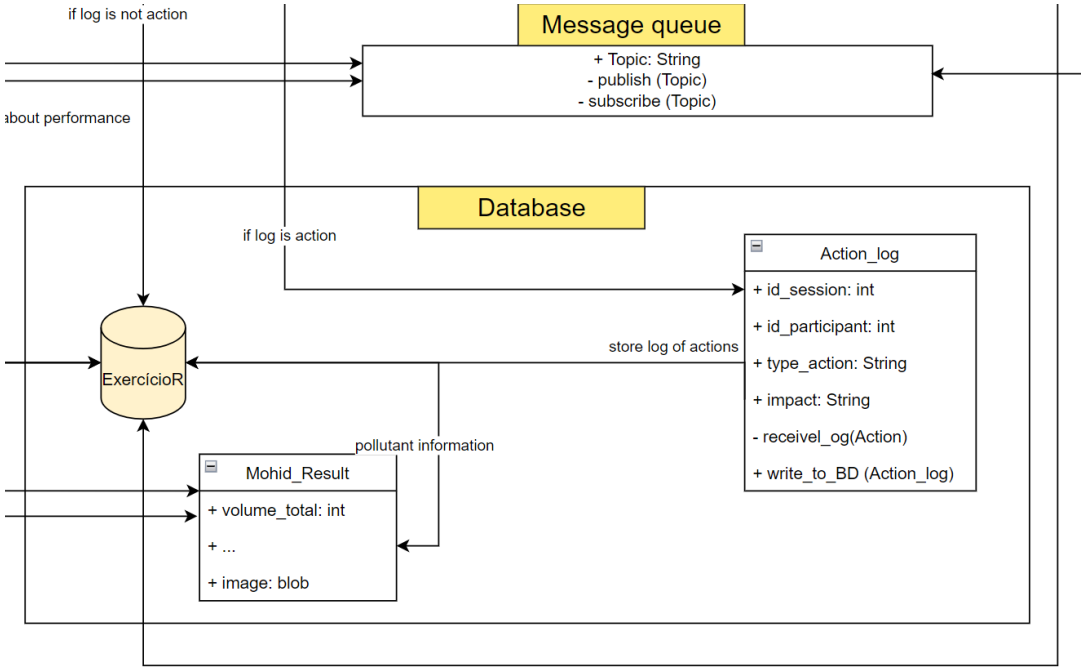

Figure 10: MPCS Message Queue and Database Interface Component

Simulation exercise configuration and game state persistence is achieved with a database component (data model on next section). The database includes also registry of MOHID results from simulation interval calculations (MOHID_Result) and log registry for tracing and analysis of exercise performance, for insights into ways to improve operational coordination.

4. MPCS Data Model

MPCS will SQL and NoSQL databases depending on the typology of data to be stored and performance decisions), considering:

- Equipment catalog and respective properties (technical specifications)
- Maritime Pollution Combat Entities (identification, responsibilities, competencies, functions, properties, technical specifications)
- MPCS parameterization (menus, listings, help, messages, interfaces, etc., in Portuguese, Spanish and English)
- Exercise actions performed and respective logging process
- Hydrodynamics and meteorology data interface with MOHID simulator

The system stores data from the exercises performed over time for KPI performance analysis and possible training impacts. There is no indication for a maximum number of stored exercises, so it is assumed that appropriate cloud infrastructure can scale to provide the necessary space and performance for the purpose.

Estimates can be generated from recordings of first exercises to allow for the selection and configuration of a server to the needs of the simulation. Storage of unstructured data (reports, images, message recordings, etc) can be more variable. Configuring the location of unstructured files (in independent cloud storage) will avoid database storage congestion.

Note on storage and use of personal data relating to the exercises:

- Encryption of unstructured files that may contain real information on participants or operations should be foreseen. The same applies to personal data, stored on any medium (eg cloud, imported for an exercise, storage on local devices).
- In case of importing data from other systems, it will be preferable to remove all personal data after completion of the exercise – provide for an encryption solution if personal data (only those strictly necessary and which cannot be obtained by API from their official source) have to be maintained for the purposes of historical simulations or performance analysis.
- Unless otherwise specified, no personal data should be stored in this system, preferring consultation, if necessary, via API to external systems that keep this record up to date and its history.

The data model for the run-time is presented on the next figure.

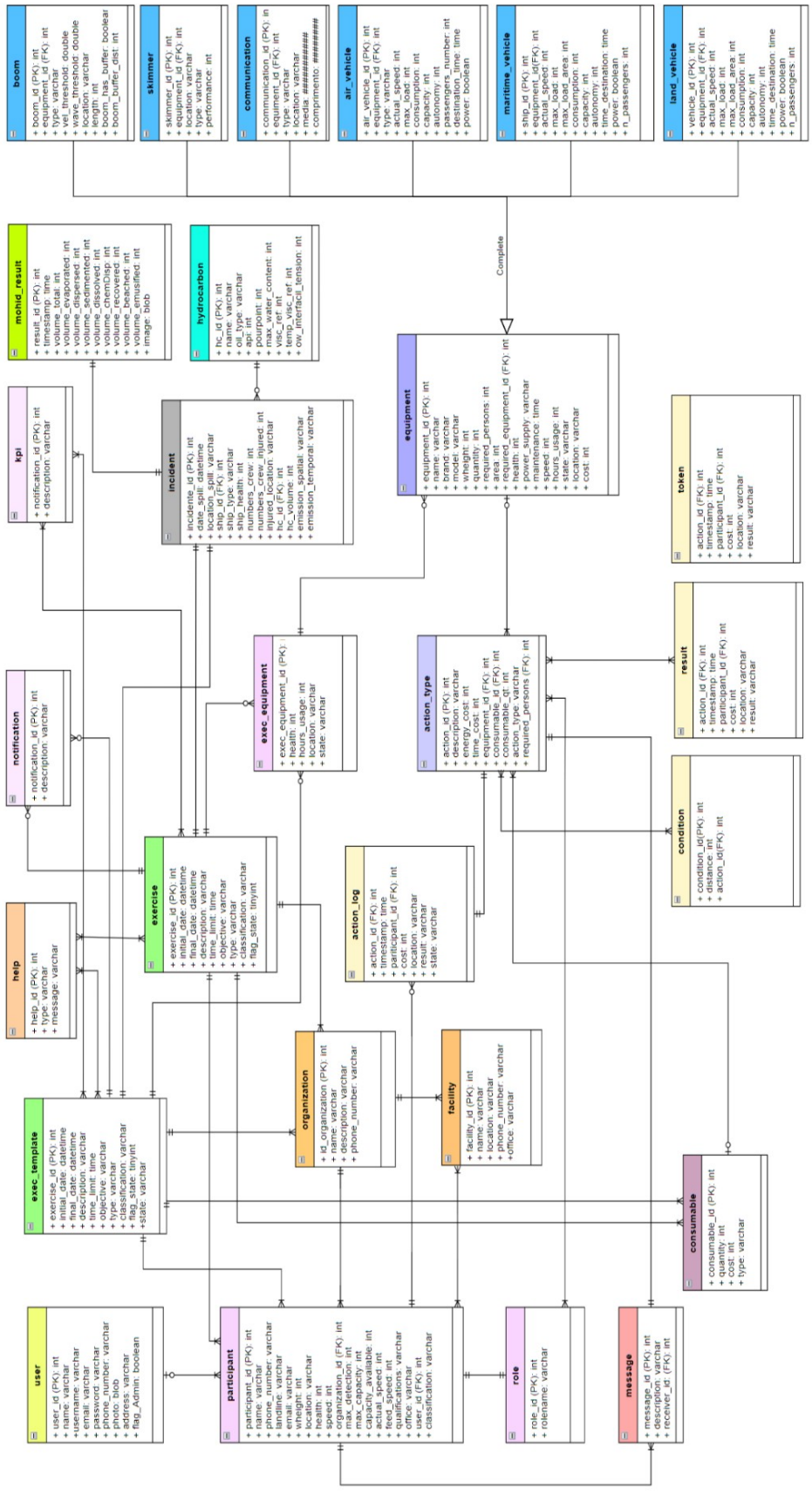

Figure 11: Data Model

5. MPCS Data Model Description

Table 2: Role

Role		
Description	Table referring to the roles that a user can have.	
Field	Type	Description
role_Id	int (primary key)	table primary key
rolename	varchar	paper name

Table 3: Users

User		
Description	Table referring to users.	
Field	Type	Description
user_Id	int (primary key)	user id
username	varchar	Logname
name	varchar	username
email	varchar	user email
password	varchar	user password
phone number	varchar	user's phone number
photo	blob	user photo
address	varchar	user address
flag_admin	boolean	flag that represents whether or

		not the user is an administrator.

Table 4: Exercise

Exercise		
Description	Table representing the exercise created by the MPCS Manager	
Field	Type	Description
exercício_Id	int (primary key)	table primary key
inicial_date	datetime	date and time the exercise should start
final_date	datetime	date and time the exercise should end
description	varchar	brief description of the exercise
type	varchar	type of exercise (training or teaching)
time_limit	time	exercise duration
objective	varchar	ex: collect as much oil as possible in x hours of exercise
classification	varchar	qualitative classification of the exercise
flag_state	tinyint	exercise state (e.g. running, draft, etc)

Table 5: Exec_Template

Exercise		
Description	Template of the table representing the exercise created by the MPCS Manager	
Field	Type	Description
exercício_Id	int (primary key)	table primary key
inicial_date	datetime	date and time the exercise should start
final_date	datetime	date and time the exercise should end
description	varchar	brief description of the exercise
type	varchar	type of exercise (training or teaching)
time_limit	time	exercise duration
objective	varchar	ex: collect as much oil as possible in x hours of exercise
classification	varchar	qualitative classification of the exercise
flag_state	tinyint	exercise state (e.g. running, draft, etc)
state	varchar	exercise state

Table 6: Incident

Incident		
Description	Event that gives rise to the stroke.	
Field	Type	Description
incident_id	int (primary key)	incident primary key
date_spill	datetime ou timestamp	date and time of occurrence of the exercise
location_spill	varchar	incident location
ship_id	int (foreign key)	id of the vessel that spilled the oil
ship_type	varchar	vessel type (ship, boat)
ship_health	int	vessel health (%)
numbers_crew	int	Number of crew on the vessel that gave rise to the incident
numbers_crew_injured	int	Number of injured crew members of the vessel
injured_location	varchar	location of injured crew members
hc_id	int (foreign key)	id referring to spilled hydrocarbon
hc_volume	int	HC volume (required attribute for Mohid)
emission_spatial	varchar	Stroke type: accident; (required attribute for Mohid)
emission_temporal	varchar	Issue type: ● instant

		• at a fixed point • along a path (required attribute for Mohid)

Table 7: Organization

Organization		
Description	Table referring to the organizations that participate in the response to the fight against pollution.	
Field	Type	Description
id_organization	int (primary key)	table primary key
name	varchar	organization name
description	varchar	organization description
phone_number	varchar	organization contact

Table 8: Facility

Facility		
Description	Table referring to facilities (buildings or group of buildings)	
Field	Type	Description
facility_id	int (primary key)	table primary key
name	varchar	installation name
location	varchar	installation location coordinates
phone_number	varchar	installation contact
office	varchar	office address or designation

Table 9: Participant

Participant		
Description	People involved in combating pollution (identity that the player in the course of removal)	
Field	Type	Description
participant_id	int (primary key)	table primary key
facilitie_id	int (foreign key)	foreign key that allows linking to the facilities table
organization_id	int (foreign key)	foreign key that allows linking to the organization table
user_id	int (foreign key)	foreign key that allows identifying the user
name	varchar	organization name
phone_number	varchar	mobile number
landline	varchar	landline phone number
email	varchar	person's email
weight	int	person's weight
location	varchar	person's location
health	int	person's health
speed	int	displacement speed (global constant of the simulator)
max_detection	int	maximum detection distance from other people and equipment
max_capacity	int	maximum capacity a person

		can carry
capacity_available	int	maximum capacity a person can carry
actual_speed	int	person's effective speed
feed_speed	int	average feeding speed
qualifications	varchar	qualifications that the person has (e.g. driving license)
office	varchar	person's office
classification	varchar	individual classification obtained in the exercise

Table 10: Equipment

Equipment		
Description	Equipment present in combating pollutant removal	
Field	Type	Description
equipment_id	int (primary key)	equipment primary key
name	varchar	equipment name
brand	varchar	equipment brand
model	varchar	equipment model
weight	int	equipment weight
quantity	int	amount existing in the fiscal year
required_persons	int	people necessary for its use

area	int	area occupied by the device
requred_equipment_id	int (foreign key)	Equipment needed to use it (assuming it's just one)
health	int	equipment health
power_supply	varchar	equipment power supply
maintenance	time	Under maintenace
speed	int	Equipment movement speed
hours_usage	int	Number of hours of equipment use
state	varchar	If the status of the equipment is: ● Normal ● Degraded ● Lost
location	varchar	equipment location
Cost	int	Cost of using equipment in euros (per hour)

Table 11: Land Vehicle

Land_Vehicle		
Description	Table representing land vehicles. This table derives from the equipment table.	
Field	Type	Description
vehicle_id	int (primary key)	table primary key
equipament_id	int (foreign key)	Foreign key referring to the

		equipment table
actual_speed	int	vehicle speed
max_load	int	corresponds to the maximum load that the land vehicle can support
max_load_area	int	corresponds to the maximum area that the load can occupy inside a vehicle
consumption	int	vehicle consumption
capacity	int	capacity of the fuel that the vehicle supports
autonomy	int	vehicle autonomy
time_destination	time	represents the time it takes the vehicle to reach the destination point
power	boolean	whether the vehicle is on or off
n_passengers	int	number of passengers the vehicle supports

Table 12: Maritime Vehicle

Maritime_Vehicle		
Description	Marine vehicle table. This table derives from the equipment table.	
Field	Type	Description
ship_id	int (primary key)	table primary key
equipment_id	int (foreign key)	Foreign key referring to the equipment table

actual_speed	int	vessel speed
max_load	int	maximum load capacity
max_load_area	int	maximum area the load can occupy
consumption	int	vehicle consumption
capacity	int	fuel capacity that the vehicle supports
autonomy	int	vessel autonomy
time_destination	time	time it takes for the vessel to reach the destination
power	boolean	vehicle on or off
n_passengers	int	number of passengers the vessel supports

Table 13: Air Vehicle

Air_Vehicle		
Description	Air vehicles used (air means)	
Field	Type	Description
air_vehicle_id	int (primary key)	air vehicle table primary key
equipment_id	int (foreign key)	foreign key
type	varchar	vehicle type
actual_speed	int	vehicle speed
max_load	int	maximum load capacity
consumption	int	vehicle consumption

capacity	int	fuel capacity
autonomy	int	vehicle autonomy
passengers_number	int	number of passengers
destination_time	time	vehicle time to destination
power	boolean	vehicle on or off

Table 14: Notification

Notification		
Description	Year start notification	
Field	Type	Description
notification_id	int (primary key)	notification primary key
description	varchar	text present in the notification

Table 15: Message

Message		
Description	Messages exchanged between users throughout the exercise	
Field	Type	Description
message_id	int (primary key)	message primary key
receiver_id	int (foreign key)	secondary key that indicates the participant receiving the message
description	varchar	message content

Table 16: Hydrocarbon

Hydrocarbon		
Description	Existing hydrocarbons	
Field	Type	Description
hc_id	int (primary key)	hydrocarbon primary key
name	varchar	hydrocarbon name
oil_type	varchar	type of hydrocarbon (e.g. crude or refined)
api	int	API Grade Value - American Petroleum Institute
pourpoint	int	Temperature (°C) at which the hydrocarbon loses its fluid properties
max_water_content	int	Maximum water content (value in %)
visc_ref	int	Reference dynamic viscosity (in cP)
temp_visc_ref	int	Temperature (°C) at which the reference dynamic viscosity was determined
ow_interfacil_tension	int	Water-oil interfacial tension (Dyne/cm)

Table 17: Boom

Boom		
Description	existing barriers	
Field	Type	Description
booms_id	int (primary key)	barrier primary key
equiment_id	int (foreign key)	foreign key that allows inheritance with the equipment table
type	varchar	barrier type
vel_threshold	double	Current speed beyond which the barrier cannot contain particles
wave_threshold	double	Significant wave height beyond which the barrier cannot contain particles
location	varchar	String that contains the vertices of one that defines the position of the barrier
length	int	barrier size
boom_has_buffer	boolean	# Boolean value (0-false or 1-true) that defines whether the barrier has a tolerance distance from which particles can be trapped in the barrier
boom_buffer_dist	int	Tolerance distance (in meters) from which particles can be trapped in the barrier

Table 18: Skimmer

Skimmer		
Description	existing skimmers	
Field	Type	Description
skimmer_id	int (primary key)	table primary key
equiment_id	int (foreign key)	foreign key to equipment table
location	varchar	skimmer location. It's just a dot on the map
type	varchar	type of skimmer
performance	int	corresponds to the performance of the skimmer

Table 19: Consumable

Consumable		
Description	Consumables that players or equipments should consume	
Field	Type	Description
consumable_id	int (primary key)	primary key of the table
quantity	int	quantity of the consumable
cost	int	cost of the consumable
type	varchar	tipo de consumível

Table 20: Communication

Communication		
Description	Communications done in the system	
Field	Type	Description
communication_id	int (primary key)	primary key of the table
equipment_id	int (foreign key)	foreign key to the equipment table
type	varchar	Voice, Text, etc
location	varchar	location where the communication is done
media	varchar	type of media exchanged
range	int	Communication distance reach

Table 21: Help

Help		
Description	Help provided by the game during the exercise	
Field	Type	Description
help_id	int (primary key)	help primary key
type	varchar	kind of help
message	varchar	help content

Table 22: Mohid_Result

Mohid_Result		
Description	Table with the result of executing the MOHID for a timestamp	
Field	Type	Description

result_id	int (primary key)	result primary key
volume_total	int	Total volume of HC
volume_evaporated	int	evaporated volume
volume_dispersed	int	dispersed volume
volume_sedimented	int	Sedimented volume
volume_dissolved	int	dissolved volume
volume_chemDisp	int	Volume removed by chemical dispersion
volume_recovered	int	Mechanically removed volume
volume_beached	int	Volume that reached the coast
volume_emulsified	int	(Appears in the described document as an output but does not appear in the file)
timestamp	time	update time
image	blob	graphical representation of the stain

Table 23: Action_type

Action_type		
Description	Represents the actions that a participant can do (sleep, eat, rest, etc…)	
Field	Type	Description
action_id	int (primary key)	primary key of the table
equipment_id	int (foreing key)	foreign key to connect to equipment table

consumable_id	int (foreign key)	Foreign key to connect to consumable table
description	varchar	description of the action
energy_cost	int	energy used to do the action
time_cost	int	time of the action
action_type	varchar	type of action
consumable_qt	int	consumable quantity needed to perform the action
required_persons	int (foreign key)	required persons for the action

Table 24: Action_log

Action_log		
Description	Action log table	
Field	Type	Description
action_id	int (foreign key)	foreign key to call the Table action
participant_id	int (foreign key)	who participated in the action
timestamp	time	duration of action
cost	int	cost of action
location	varchar	where the action took place
result	varchar	what the action results in
state	varchar	state of action (plan, avail, started, done)

Table 25: Condition

condition		
Description	Summary of conditions to be verified on action entry	
Field	Type	Description
action_id	int (foreign key)	Foreign key to call the Table action_type …_log
distance	int	Minimum distance to target object of action or equipment
material	int	code for required material
persons	int	number of persons required
energy	int	energy source required

Table 26: Result

result		
Description	Impact/Result of an action	
Field	Type	Description
result_action_id	int (primary key)	Foreign key to call the Table action_type …_log
action_id	int (foreign key)	Foreign key to call the Table action_type …_log
material	int	material output from action
persons	int	persons available after action
energy	int	energy available after action

Table 27: Token

token		
Description	Token table will serve to save the Petri Net marking recording the game state	
Field	Type	Description
token_id	int	key serves as token type
name	varchar	what is - human readable desc
quantity	int	amount of token available

Table 28: Kpi

kpi		
Description	Table to process and register current calculation of key performance indicators	
Field	Type	Description
kpi_id	int	indicator key
kpi_name	varchar	human readable designation
kpi_value	varchar	textual value encoding
kpi_func	varchar	javascript function ref

6. Development Technologies

The MPCS Software System Architecture will be developed using the following technologies.

Browser-based Internet Technologies

MPCS Simulator User Interfaces will be accessible through a web browser, using web technologies such as HTML, CSS, and JavaScript. The interface's structure and content are provided by HTML, and its style and aesthetic appeal are created by CSS. The interface is made more interactive and dynamic with JavaScript. A browser-based user interface is a common approach to creating software applications that run on different platforms and devices, including desktops, laptops, tablets, and smartphones. By choosing a browser-based approach for the development of the User Interfaces, the use of the MPCS Simulator will be more accessible, allowing its exploration in different contexts of use and scenarios.

Spring Boot

The MPCS Simulator backend will be developed with the Spring Boot framework. Following a microservices architecture, Spring Boot is a Java-based framework that optimises the process of building and deploying web applications, providing a flexible way of building enterprise-level applications. Spring Boot also provides a range of built-in features and libraries, including Spring Data for database access, Spring Security for authentication and authorization, and Spring MVC for building RESTful web services, which will help in the integration of the MPCS Simulator components.

MySQL

Data management in the MPCS Simulator will be supported by MySQL technology. MySQL is a popular open-source relational database management system that provides a flexible way to store, manage, and retrieve data, supporting a wide range of data types, including text, numeric, date/time, and binary data. In order to guarantee data integrity, MySQL's support for transactions enables several operations to be combined into a single atomic unit of work. This is a key feature of the MPCS Simulator in order to support multi-user activities, including both synchronised and concurrent actions.

RabbitMQ

The communication between the MPCS Simulator components will be supported by RabbitMQ software. The open-source message broker RabbitMQ offers applications a flexible and dependable means to exchange messages. RabbitMQ uses a publisher-subscriber model, where producers send messages to a queue, and consumers retrieve messages from the queue, allowing them to handle a large volume of messages. This is a key feature of the MPCS Simulator in order to allow the concurrent activity of a large number of users in a non-blocking way.

Tomcat

The MPCS Simulation web applications will run on a Tomcat server. Tomcat is an open-source Java-based web server and servlet container that offers an adaptable approach to delivering Java-based web applications. One of the most popular web servers, it is renowned for its dependability and stability. It is a secure and dependable option for enterprise-level

application development thanks to its modular architecture, scalability, and large user and developer community.

Configuration Management

The development of the MPCS solution will take place on a specific server to be created in the data center of the Informatics Engineering Department of the University of Coimbra. It may involve several virtual machines for testing during the process.

The production server will be defined later by the consortium, according to the technical specification obtained in the system testing phase. It should be possible to start a new exercise on the production server without the intervention of the IT team / infrastructure managers.

7. Quality Attributes

Security

- Encryption of unstructured files that may contain real information on participants or operations should be foreseen and also encryption of personal data stored in any platform;
- No personal data should be stored in the system, using, if necessary, an API to external systems that keep the data updated and its history.

Performance

- Run multiplayer simulation exercises with 25 simultaneous users;
- The game should give an answer to every action of the user within 5 seconds (it can be an await message).

Usability

- Enable adequate time to learn (1 hour from first contact) and efficient use of role specific interfaces for users familiar with browser enabled devices (mobile and desktop).

Maintainability

- Maintenance to the MPCS server will depend on well documented operations. Although impact may be low, no guarantees can be given regarding maintenance involving changes such as:
    - Updates to the operating system, database engine, web server;
    - Software configurations (eg web server);
    - Data removal operations, without updated backups.

Recoverability

- A daily backup should be provided for the data relating to the exercises, and a longer period of time may be considered for the audit records, namely those associated with changes in the server (operating system updates, database, etc.); MPCS system should be recoverable from a backup copy with a limited time frame.

8. Conclusions

This document presents the first version of the MPCS Software System architecture that defines the Simulation Structure Design. This first model is a living document to be improved during the software development process.

Software system architecture defines the components and interconnections for the simulation system. It includes specification of library modules and subsystems required to operate the software developed such as database management systems and their data models.

This document summarises the MPCS concept, describes the initial proposal for MPCS Software System Architecture, defines the module components and interactions, the Data Modeland the data entities and their intended use. The document ends with a brief introduction of applicable quality attributes.

• References

Oracle (2007). Sun Java Enterprise System Deployment Planning Guide: Service Level Requirements, [online] Available: https://docs.oracle.com/cd/E19636-01/819-2326/819-2326.pdf.

Humphrey, W. S., & Starrett, E. (2009). Systems Engineering Technical Review (SETR): Navys Acquisition and Development Process and Software Lessons Learned, [online] Available: https://apps.dtic.mil/sti/pdfs/AD1014731.pdf