**TABLE OF CONTENTS**

**1** **Introduction**

***1.1*** ***Document overview***

***1.2*** ***Abbreviations and Glossary***

1.2.1 Abbreviations

1.2.2 Glossary

***1.3*** ***References***

1.3.1 Project References

1.3.2 Standard and regulatory References

***1.4*** ***Conventions***

**2** **Architecture**

***2.1*** ***Architecture overview***

***2.2*** ***Logical architecture overview***

2.2.1 Software Component 1 description

2.2.2 Software Component 2 description

2.2.3 Software Component 3 description

***2.3*** ***Physical architecture overview***

2.3.1 Hardware Component 1 description

2.3.2 Hardware Component 2 description

2.3.3 Hardware Component 3 description

***2.4*** ***Software COTS***

**3** **Dynamic behaviour of architecture**

***3.1*** ***Workflow / Sequence 1***

***3.2*** ***Workflow / Sequence 2***

**4** **Justification of architecture**

***4.1*** ***System architecture capabilities***

***4.2*** ***Network architecture capabilities***

***4.3*** ***Risk analysis outputs***

***4.4*** ***Human factors engineering outputs***

**5** **Requirements traceability**

# Introduction

## Document overview

This document describes the architecture of RALPH WINDOWS system.

It describes:

**“**Repara-FÉLIX JR” es un videojuego clásico donde lucha un villano llamado Ralph con un héroe llamado Félix. El entorno en el que se desarrolla el juego es un edificio residencial donde Ralph rompe los vidrios del edificio y Félix empieza una lucha entre las habilidades de construcción que posee.

Pero es interesante hacer una versión nueva de este juego, **“ RALPH WINDOWS ”**, es una versión alternativa en donde el objetivo es reparar la totalidad de las ventanas rotas del edificio, en este caso los participantes (jugadores) van a ser Felix (quién es el que repara las ventanas).  
  
Existirán dos equipos, equipo Félix 1 y equipo Félix 2, los cuales competirán por reparar la mayor cantidad de ventanas y entre equipos podrán colaborarse, se sabe qué equipo fue el ganador dependiendo de la cantidad de ventanas que repare.

## Abbreviations and Glossary

### Abbreviations

Equipo Felix 1: Equipo 1

Equipo Felix 2: Equipo 2

API : Application Programming Interface

STOMP : Streaming Text Oriented Messaging Protocol

### Glossary

### REST: las operaciones más importantes relacionadas con los datos en el sistema REST y las especificaciones HTTP que se utilizaran son cuatro :

### POST(crear)

### Get(Leer)

### PUT(editar)

### DELETE( eliminar)

* API: es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
* STOMP: Es un protocolo basado en texto, diseñado para trabajar con MOM. Proporciona un formato interoperable que permite a clientes STOMP hablar con cualquier mensaje broker que soporta el protocolo.

## References

### Project References

|  |  |  |
| --- | --- | --- |
| # | Document Identifier | Document Title |
| [R1] | http://juegos.disney.es/rompe-ralph/repara-felix-jr | Referencias del juego original |
| [R2] | http://webgenio.com/2017/01/14/40-bonitos-formularios-css-registros-login-web/ | Plantillas html y css ingreso de usuarios |

### Standard and regulatory References

|  |  |  |
| --- | --- | --- |
| # | Document Identifier | Document Title |
| [STD1  ] |  | Add your documents references.  One line per document |

## Conventions

No aplica.

# Architecture

## Architecture overview

**Perspectiva de usuario:**

Este es un software el cual se desarrolló para un juego con el fin de que cualquier persona lo pueda usar ya sea apasionado por los videojuegos o no.

Al iniciar el juego el jugador debe ingresar con su nombre de usuario y podrá elegir el color del equipo al cual desea pertenecer, el juego no inicia hasta que estén todos los jugadores registrados (en este caso es 1 jugador por equipo), es decir, el jugador debe esperar hasta que ambos equipos se encuentren con el número completo de jugadores. Cuando el último jugador ingrese el juego inicia.

Cada uno de los jugadores independientemente del equipo al que pertenecen se pueden mover hacia cualquier lado (arriba, abajo, derecha, izquierda) del mapa del juego, con el fin de arreglar las ventanas del edificio y de esta manera poder ganar el juego. Gana el juego el equipo que arregle una mayor cantidad de ventanas.

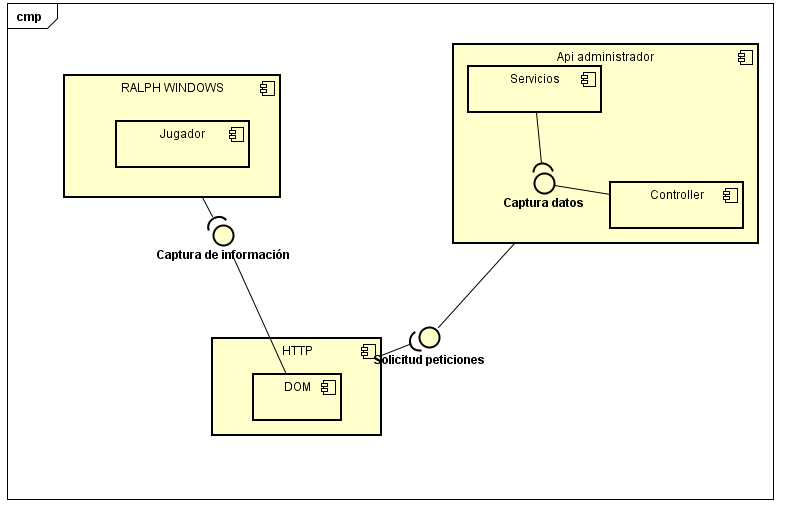
Nuestra aplicación maneja dos interfaces: la del jugador el cual se encarga de conocer la información de los jugadores y la del mapa de juego en donde se evidenciaran los movimientos de los equipos.

**Perspectiva del desarrollador:**

Esta aplicación fue desarrollada bajo los principios de API-REST, la cual se apoya en el protocolo HTTP en donde cualquier cliente con la misma URL se podrá comunicar y podrá acceder a las páginas relacionadas al juego RALPH WINDOWS, cuando el usuario acceda al recurso podrá usar cualquier servicio del juego con las respectivas operaciones, haciendo uso de GET, POST , PUT , DELETE.

Se manejaron conceptos de arquitectura STOMP y webSockets, con el fin de que la aplicación pudiera establecer una comunicación entre el cliente y el servidor, logrando acciones en tiempo real desde varios equipos, esto se lleva a cabo haciendo uso de un mediador de mensajes que van conectados a operaciones de la aplicación, dicha interacción se hace mediante “Connect, subscribe, send, Disconnect”.

## Logical architecture overview



### Software Component 1 description

**Ralph Windows:** Nuestra aplicación requiere de un cliente que será aquel que ingrese como usuario, se almacena y se captura la información al DOM que será aquel que cargara estos datos para asignar un mapa de juego.

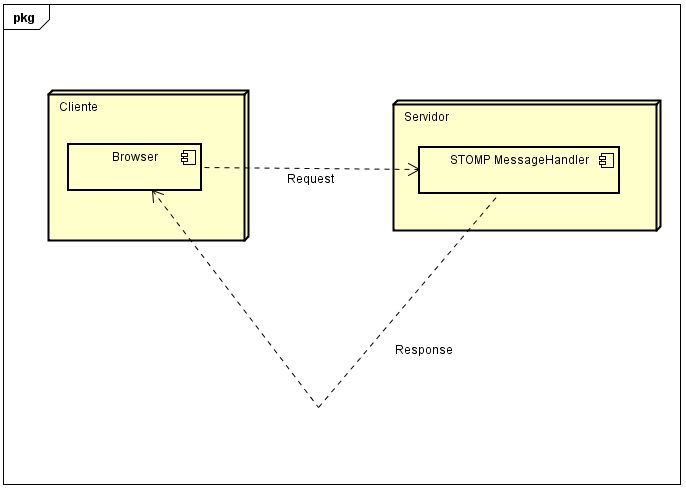
### 2.2.2 Software Component 2 description

**Services:** Se encuentran las clases que tiene la labor de implementar las operaciones internas que hacen las peticiones HTTP para que el controlador solo requiere llamar estas actividades y devolver la información al JavaScript , logrando una comunicación entre la parte lógica mediante métodos respectivos.

**2.2.3 Software Component 3 description**

**Controller:** Estan las clases encargadas de implementar las operaciones que se han solicitado desde el javascript para establecer una comunicación y hacer un manejo de los datos, para poder mostrarlos en el html.

## Physical architecture overview



### Hardware Component 1 description Cliente: También conocido como usuario o jugador, será la persona que ingrese al juego y pueda interactuar con otras personas las cuales podrán pertenecer a su mismo equipo o al equipo rival. En este arquitectura el cliente realiza peticiones al servidor y el servidor es aquel que generará una respuesta.

### Hardware Component 2 description

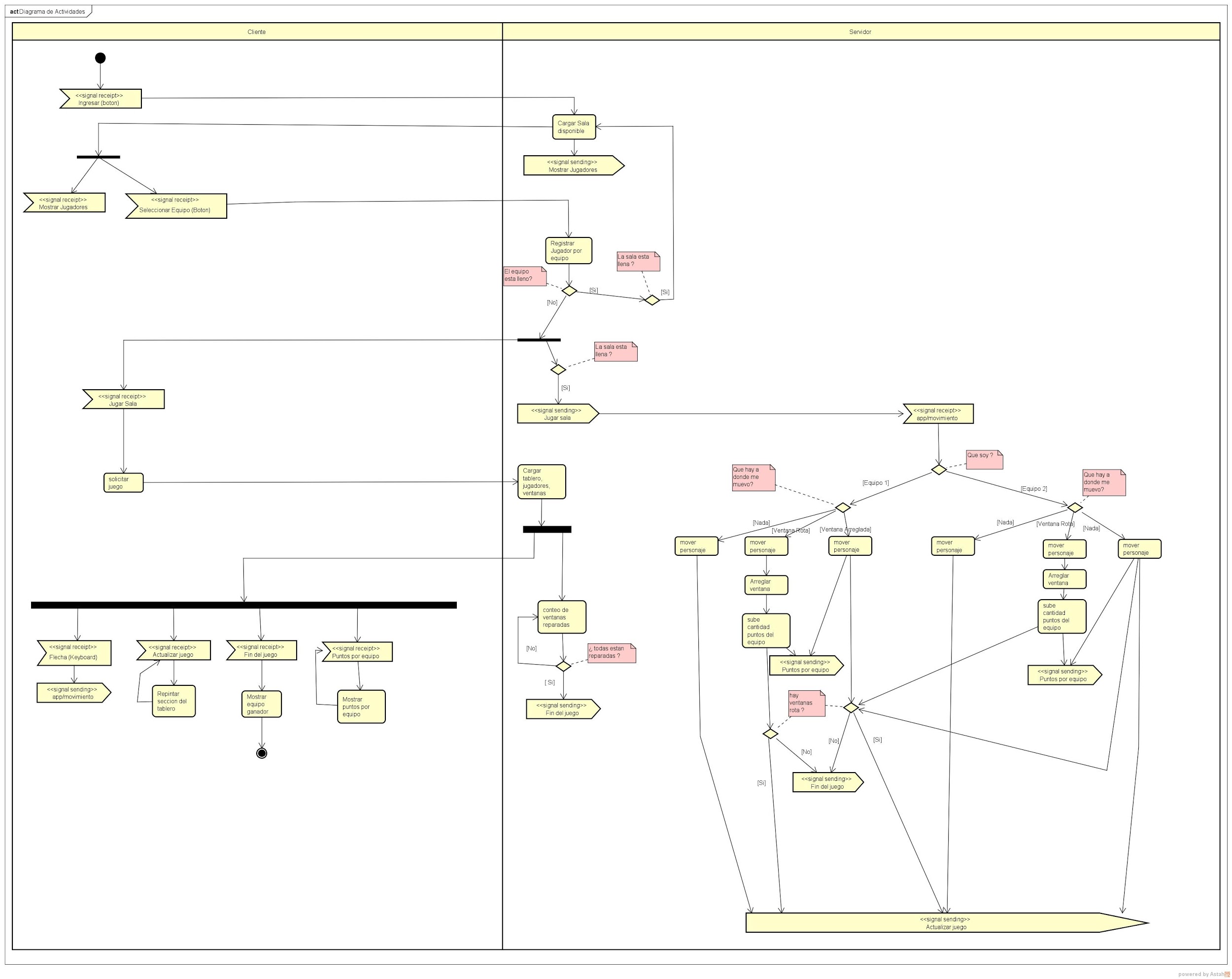
Servidor: Es aquel que va a generar la respuesta a solicitudes realizadas por el cliente,

desde el momento de iniciar el juego hasta su final.

## Software COTS No aplica.

# Dynamic behaviour of architecture

## Workflow / Sequence 1



## Workflow / Sequence 2

Repeat the patern for each main function of the system

# Justification of architecture

## System architecture capabilities

Describe here the rationale of the hardware / software architecture in terms of capabilities :

* Performances (for example response time, user mobility, data storage, or any functional performance which has an impact on architecture)
* User / patient safety (see §4.3 and §4.4)
* Protection against misuse (see 4.4)
* Maintenance (cold maintenance or hot maintenance),
* Adaptability, flexibility
* Scalability, availability
* Backup and restore
* Hardware and Software security : fault tolerance, redondancy, emergency stop, recovery after crash …
* Administration,
* Monitoring, audit
* Internationalization

## Network architecture capabilities

If the medical device uses/has a network, describe here the rationale of the hardware / network architecture :

* Bandwidth
* Network failures
* Loss of data
* Inconsistent data
* Insconsistent timing of data
* Cyber security (see FDA Guidance on Cyber Security of networked medical devices)

## Risk analysis outputs

If the results of risk analysis have an impact on the architecture, describe here for each risk analysis output what has been done to mitigate the risk in the architecture.

Use diagrams if necessary, like architecture before risk mitigation and architecture after risk mitigation, to explain the choices.

## Human factors engineering outputs

If the results of human factors analysis have an impact on the architecture, describe here for each risk human factors output what has been done to mitigate the risk in the architecture.

# Requirements traceability

Add a table with traceability of components of this document with functional requirements.

|  |  |  |
| --- | --- | --- |
| Requirement | Component | Comment |
| REQ-001  The device shall do foo | COMPO-001: foo maker | COMP-001 does foo.  COMP-002 also does verification of foo. |

This may be a difficult job. A high level function is usually handled by many components. In this case, quote only the component(s) which has(have) the major role.