UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Tecnologías de la Información

TRABAJO FIN DE GRADO

QuizzIS: Sistema de Gestión para un Concurso Televisivo

Luis Miguel Ortiz Rozalén

Julio, 2019

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

Tecnologías de la Información

TRABAJO FIN DE GRADO

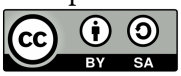
QuizzIS: Sistema de Gestión para un Concurso Televisivo

Autor(a): Luis Miguel Ortiz Rozalén

Director(a): Carlos González Morcillo

Julio, 2019

QuizzIS: Sistema de Gestión para un Concurso Televisivo



Tribunal:

Presidente: _____

Vocal: _____

Secretario: _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario
Fdo.:	Fdo.:	Fdo.:

A todas las máquinas von Neumann con patas
que he conocido durante todos estos años

Resumen

En la Escuela Superior de Informática de Ciudad Real, se lleva a cabo periódicamente un evento en el que varios equipos participantes responden una serie de preguntas para obtener el máximo número de puntos. El objetivo de este evento es promover y fomentar la Informática entre los estudiantes de Educación Secundaria de Castilla-La Mancha. Cada vez que se celebra este evento, se debe realizar la renovación y configuración del sistema utilizado para gestionar el concurso.

Por lo tanto, este Trabajo de Fin de Grado ha surgido como una nueva solución a este problema. En esta renovación se ha mejorado la simplicidad y robustez del sistema anterior, con el uso de tecnologías básicas y actuales. Sin embargo, el nuevo software conserva la misma funcionalidad que el anterior.

QuizzIS consiste en un sistema distribuido con arquitectura cliente-servidor que da soporte al ciclo de vida completo de cualquier concurso; permite la secuenciación de preguntas multimedia, habilita la integración con los equipos de producción disponibles, es capaz de desplegar contenido audiovisual, proporciona una interfaz de control y da soporte a la participación del público a través de dispositivos móviles.

Abstract

A competitive event is periodically held at Escuela Superior de Informática in Ciudad Real where several participating teams answer a set of questions in order to obtain the maximum score. The aim of this contest is to promote and foment IT among students of Secondary Education of Castilla-La Mancha. Whenever this competition is celebrated, the renewal and configuration of the system used to manage the contest must be accomplised.

Therefore, this Final Degree Project has emerged as a new solution to this issue. In this renovation, the simplicity and robustness of the previous system has been improved, with the use of basic and current technologies. Despite the changes and improvements executed, the software maintains the same functionality as the previous one.

QuizzIS consists of a distributed system with a client-server architecture supporting the entire life cycle of any contest; it allows the sequencing of multimedia questions, enables integration with available production equipment, is able to display audiovisual content, provides a control interface and supports external public engaging through mobile devices.

AGRADECIMIENTOS

«La gratitud es una flor que brota del alma.»
Henry Ward Beecher

Luis Miguel Ortiz Rozalén

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

ÍNDICE DE ALGORITMOS

LISTADO DE ACRÓNIMOS

AJAX Asynchronous JavaScript And XML.

APL Arduino Programming Language.

CSS3 Cascading Style Sheets 3.

ESI Escuela Superior de Informática.

GNU GNU's Not Unix.

HDMI High-Definition Multimedia Interface.

HTML5 HyperText Markup Language 5.

IDE Integrated Development Environment.

JSON JavaScript Object Notation.

MAMP Mac OS X Apache MySQL PHP.

MPEG-4 Moving Picture Experts Group 4.

NoSQL Not Only Structured Query Language.

PHP PHP Hypertext Preprocessor.

TFG Trabajo de Fin de Grado.

UCLM Universidad de Castilla-La Mancha.

USB Universal Serial Bus.

CAPÍTULO 1

INTRODUCCIÓN

El avance de la Informática a lo largo de los últimos años ha logrado que la tecnología esté presente en muchos ámbitos de nuestras vidas. Por este motivo, es de gran importancia que los estudiantes conozcan este campo de conocimiento.

Hay muchas instituciones educativas que unen a la iniciativa de organizar un evento dedicado a incentivar y promover la Informática entre los estudiantes de Educación Secundaria. Este evento generalmente se denomina *Olimpiada Informática*, y consiste en una serie de pruebas que los estudiantes deben ir superando.

La fase final de la Olimpiada Informática normalmente se realiza de forma presencial. En la ESI de Ciudad Real este evento sigue la misma dinámica que un concurso televisivo. Por esta razón, es necesario contar con un sistema encargado de gestionar la mecánica completa del concurso.

El presente TFG ha sido la renovación y configuración del sistema utilizado en las ediciones anteriores de la fase final [Mar13]. Por lo tanto, *QuizzIS* es un sistema de gestión para un concurso televisivo con soporte para participación del público.

1.1. OLIMPIADA INFORMÁTICA

La Olimpiada Informática es una actividad que busca promover el interés por la Informática entre los estudiantes, presentando una visión amplia de la misma y alentando a los jóvenes a convertirse en futuros ingenieros [CIT+18]. También es objetivo primordial fomentar las buenas prácticas en el aprendizaje de la Informática en el nivel preuniversitario.

La UCLM por su parte organiza anualmente su propia Olimpiada Informática, en la cual participan gran cantidad de Institutos de Educación Secundaria de la región. Cada año se realiza alternativamente entre Ciudad Real y Albacete. En cada edición los alumnos participantes deben pertenecer al nivel de estudios de Bachillerato o de Ciclos Formativos.

Entre enero y abril de este año, cada uno de los equipos han realizado tres pruebas *on-line*. En cada prueba, los temas tratados han sido diferentes y se han adaptado a las materias

enseñadas en sus estudios. El pasado 24 de mayo se ha llevado a cabo en Ciudad Real la fase final. Los tres equipos clasificados en cada categoría han competido de la misma manera que lo harían en un concurso de televisión.

1.2. CONCURSOS TELEVISIVOS

La televisión es fundamentalmente entretenimiento. Hoy en día, el espectáculo de televisión está identificado en gran medida con la *telebasura*. Sin embargo, entretener con contenido de calidad puede asumir tareas tan importantes como las de informar o educar. Entre la inmensa cantidad de formatos de entretenimiento, el concurso se distingue por transmitir valores como el afán de superación, la divulgación de conocimientos y su capacidad para involucrar a la audiencia en el programa.

Con la inclusión de este formato en la fase final de la Olimpiada, se crea una atsmósfera que consigue cautivar la atención tanto de los concursantes como del público. Además, a través de este tipo de actividades los estudiantes perciben el aprendizaje de la Informática como una tarea divertida, que gradualmente ayuda a crear vínculos afectivos con este área de conocimiento.

Esta solución, por tanto, demuestra que puede ser extendida a cualquier concurso televisivo con tan sólo la personalización de la temática de las preguntas.

1.3. NECESIDAD DE UN SISTEMA DE GESTIÓN

Con el trascurso de las ediciones la Olimpiada, surge la necesidad de contar con un sistema que permita tener un control completo del desarrollo del concurso. Esto supone reducir ligeramente los aspectos más automatizados de los que disponía el sistema anterior.

Antes de la producción del concurso, se deben generar las diferentes preguntas que se lanzarán el día del evento. El sistema debe permitir la reproducción de contenido multimedia, ya que cada pregunta contiene imágenes, videos y/o sonidos asociados. También requiere un mecanismo de control distribuido que permita la realización del concurso, con el envío de señales y eventos desde diferentes puntos para dirigir su desarrollo. También debe permitir el despliegue físico y la integración con el hardware disponible.

A partir de todas estas condiciones, se extraen las siguientes características que debe satisfacer el sistemga de gestión para un concurso televisivo:

- Permitir el despliegue gráfico multimedia, con posibilidad de incorporar vídeos.
- Reproducir efectos sonoros y música ambiental.

- Integración con los sistemas existentes.
- Ofrecer interfaces de control multidispositivo para la realización del concurso.
- Ofrecer mecanismos específicos de recuperación de errores.

En el capítulo 2 se explican con mayor detalle estas características. *QuizzIS* se propone como un sistema de gestión para un concurso televisivo que satisface dichas funcionalidades. Los procedimientos y las tecnologías utilizadas para su desarrollo se describen en el capítulo 3.

1.4. ESTRUCTURA DEL DOCUMENTO

Este documento se ha redactado de acuerdo a la guía de estilo y formato para TFG de la ESI de Ciudad Real, incluyendo los siguientes capítulos:

- **Capítulo 1. Introducción:** se explica el contexto y la motivación que lleva al desarrollo del proyecto.
- **Capítulo 2. Objetivo:** en este capítulo se detalla el objetivo general y los objetivos específicos que persiguen el presente trabajo.
- **Capítulo 3. Metodología:** se explican las diferentes metodologías que se han utilizado, así como las tecnologías empleadas a lo largo del desarrollo.
- **Capítulo 4. Resultados:** en este capítulo se analizan los resultados obtenidos después de la elaboración del trabajo.
- **Capítulo 5. Conclusiones:** se evalúa el trabajo realizado en base a los objetivos y se establecen futuras mejoras.

CAPÍTULO 2

OBJETIVO

En este capítulo se explican con más detalle tanto el objetivo general como el desglose de objetivos específicos.

2.1. OBJETIVO GENERAL

El objetivo general del proyecto se establece como el **desarrollo de un sistema de gestión cuya finalidad es la realización de un concurso en formato televisivo que permita la participación del público.**

QuizzIS debe permitir el control integral de la mecánica del concurso, desde la configuración de los parámetros iniciales hasta la realización en directo del mismo.

2.2. OBJETIVOS ESPECÍFICOS

A partir del objetivo principal descrito en la sección anterior, se definen los siguientes subobjetivos:

2.2.1. Despliegue gráfico multimedia

Disponer de diferentes elementos que permitan el despliegue de objetos multimedia tales como vídeos, textos e imágenes. También tener soporte para la reproducción de sonido.

El sistema debe ser capaz de:

- Procesar archivos multimedia en tiempo real, especialmente vídeos.
- Emplear el formato de fuentes *TrueType* con un tamaño proporcionado al espacio disponible en pantalla.
- Reproducir música y efectos de sonido asociados a vídeos o eventos.

2.2.2. Integración con los sistemas existentes

La versión anterior del sistema integraba una plataforma para la intervención del público, sin embargo, en la última edición del concurso no funcionó como se esperaba. En esta nueva versión, esta plataforma se desprende de *QuizzIS* y en su lugar se utiliza una herramienta ya existente. Esto hace necesario sincronizar la secuencia de vídeos entre ambos.

La herramienta en cuestión es *TurningPoint*. TurningPoint es una solución que permite la fácil participación de los estudiantes a tiempo real utilizando sus propios dispositivos.

La herramienta ofrece las mismas prestaciones que la plataforma anterior:

- Soporte para cualquier sistema operativo (Android, iOS y Windows Phone).
- Mecanismo de autenticación que permiten identificar a cada estudiante.
- *Feedback* para los usuarios, de modo que les resulte sencillo seguir el desarrollo del concurso.
- Recopilación de datos en tiempo real, haciendo posible consultar la clasificación en cualquier momento.

2.2.3. Interfaces de control multidispositivo

Desarrollar una aplicación que permita un control distribuido del sistema. Cada instancia puede enviar señales a un mismo servidor, el cual mantiene y distribuye el estado del concurso, garantizando así la coherencia.

QuizzIS debe permitir:

- Envío señales de control que dirijan el desarrollo del concurso en función de los acontecimientos.
- Ejecución múltiples instancias, de forma que se comparta la responsabilidad de control del concurso en diferentes localizaciones (presentador, técnico de control de vídeos y técnico de control de marcadores).
- Mostrar información suficiente para entender el estado del concurso y poder presentarlo de forma sencilla.

2.2.4. Sistema robusto y con mecanismos específicos de recuperación de errores

El sistema debe dar soporte para la producción de concursos en directo. Así, es importante que *QuizzIS* no falle de forma crítica y pueda recuperarse de cualquier fallo puntual, sin

afectar el desarrollo del concurso. Por lo tanto, debe ser tolerante a errores utilizando técnicas como la replicación y la recuperación automática.

Así, el sistema debe permitir:

- Replicación de clientes en otros dispositivos, de forma que pueda realizarse una sustitución rápida.
- Recuperación ante cualquier fallo en los clientes, tratando de reiniciar los procesos de forma automática.

2.2.5. Sistema distribuido y heterogéneo

QuizzIS debe estar formado por varias aplicaciones que se ejecutan en diferentes máquinas, tanto en ordenadores como en otros controladores hardware. Deben estar comunicadas entre sí por una red de comunicaciones, formando un sistema distribuido.

Por ello, el sistema debe modelar:

- Comunicación entre aplicaciones escritas en diferentes lenguajes.
- Arquitectura cliente-servidor que permita la comunicación entre aplicaciones.

2.2.6. Arquitectura flexible

Para favorecer la reutilización de *QuizzIS*, debe ser diseñado con vistas a la flexibilidad de despliegue. De esta manera, podrá ser utilizado para la gestión integral de concursos bajo diferentes condiciones.

Por eso, el sistema debe:

- Poder ser lanzado en cualquier computadora.
- Utilizar elementos compatibles con cualquier navegador.

2.2.7. Basado en estándares y tecnologías libres

El proyecto debe ser llevado a cabo utilizando fundamentalmente software y hardware libre. Por lo tanto, *QuizzIS* debe, en la medida de lo posible:

- Construirse utilizando herramientas con licencias libres, sin coste económico.
- Utilizar bibliotecas de código abierto y con licencias libres para la implementación.
- Desarrollarse empleando bibliotecas multiplataforma.

CAPÍTULO 3

METODOLOGÍA

En este capítulo se explica la metodología empleada para la planificación y el desarrollo del proyecto. También incluye el marco tecnológico en el que estuvo involucrado el proyecto.

3.1. METODOLOGÍA DE TRABAJO

El proyecto *QuizzIS* ha sido considerado un **desarrollo ágil de software**, centrado en los factores humanos y el producto software. Esto es, se da mayor valor a las personas, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas [CL12].

Se trata de una alternativa a los desarrollos de software tradicionales. Mientras que las metodologías ágiles son adaptativas y están orientadas a las personas, las tradicionales se basan en la planificación estricta y están orientadas a los procesos [Cad+13].

Todo desarrollo ágil se fundamenta en los valores recogidos en el *Manifiesto Ágil* [Bec+01]:

- **Individuos e interacciones sobre procesos y herramientas**: el proyecto se desarrolla en torno a individuos motivados. Se ponen a su disposición el entorno y el apoyo que necesitan.
- **Software funcionando sobre documentación extensiva**: el incremento de funcionalidad frente a la documentación es la medida principal para el progreso.
- **Colaboración con el cliente en la negociación contractual**: los promotores, desarrolladores y usuarios se reúnen y trabajan de forma constante.
- **Respuesta ante el cambio sobre seguir un plan:** se acepta que los requisitos cambien, incluso en las últimas etapas de desarrollo.

En los primeros años de desarrollo ágil aparecieron una gran cantidad de metodologías que abordaban en diferente grado los principios del manifesto. Entre ellas se inlcuyen

Programación Extrema (XP), Scrum, Desarrollo de Software Lean (LSD), Desarrollo Basado en Funcionalidades (FDD) y Metodologías Crystal [Din+12].

En este contexto, el intento por aplicar de forma excluyente una metodología ágil ha supuesto un obstáculo [Let13]. Por lo tanto, se ha llevado a cabo una adaptación tomando como referencia el **desarrollo iterativo e incremental** y empleando determinados rasgos de **Scrum**.

Como indica Schwaber [Sch04], Scrum está orientado hacia un proceso de control empírico. Es decir, el proceso para guiar el control del trabajo hacia el resultado más rentable. En este proceso las decisiones se basan en la observación y la experimentación. Es por esto que Scrum, junto con el proceso iterativo e incremental, aumenta la previsibilidad y el control de riesgos. De esta forma es posible afrontar proyectos de gran complejidad.

La forma en que Scrum opera se basa en tres ideas principales: **transparencia**, **inspección** y **adaptación**.

Cada iteración, denominada Sprint, dura normalmente entre 2 y 4 semanas. Al comienzo del Sprint, el equipo elabora una lista de tareas. Después, se estiman sus esfuerzos y se asignan a los diferentes miembros del equipo. El equipo trabaja por su cuenta durante el resto del Sprint, organizando reuniones diarias donde se realiza una puesta en común. A estas reuniones se les llama Daily Scrum. Al final del Sprint, el equipo presenta el incremento de funcionalidad desarrollado para que el cliente pueda inspeccionarlo y realizar las adaptaciones oportunas al proyecto [SS10].

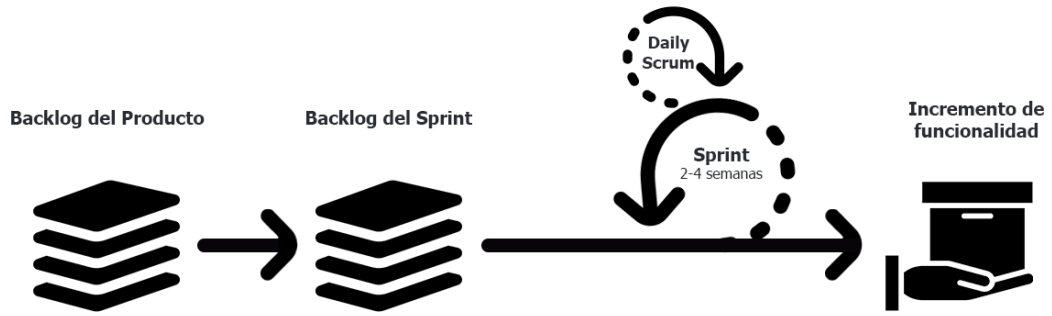


Figura 3.1: Diagrama del proceso de desarrollo Scrum.

El Backlog del Producto consiste en una lista de requisitos donde se recopilan las **historias de usuario** (Secc. 4.1). A partir de esta, se elabora la lista de tareas mencionadas

anteriormente, conocida como el Backlog del Sprint.

Las responsabilidades en un proyecto basado en Scrum se dividen en **tres roles** [SS10]:

- **Product Owner**: es quien conoce las necesidades del cliente y el encargado de reflejarlas en el Backlog del Producto. También tiene la responsabilidad de financiar el proyecto de principio a fin.
- **Scrum Master**: es el responsable del proceso Scrum. Su labor es revisar e inspeccionar la adaptación de los cambios, y proporcionar el *feedback* necesario al equipo de desarrollo para lograr los objetivos definidos.
- **Development Team**: son los responsables del desarrollo del producto. El equipo es autogestionado, autoorganizado y multifuncional. En cada iteración deben convertir el Product Backlog en un incremento de funcionalidad. Tienen la responsabilidad de que cada iteración, y el proyecto en general, salga adelante.

Para este **caso concreto**, la metodología ha sido adaptada a las necesidades del equipo y del proyecto. De esta forma, las iteraciones han durado de 1 a 5 semanas, realizando una reunión semanal que ha facilitado la comunicación y la colaboración entre los diferentes miembros del proyecto. Sin embargo, en cada iteración se mantiene el incremento de la funcionalidad, con el objetivo de lograr los requisitos definidos al comienzo de la iteración.

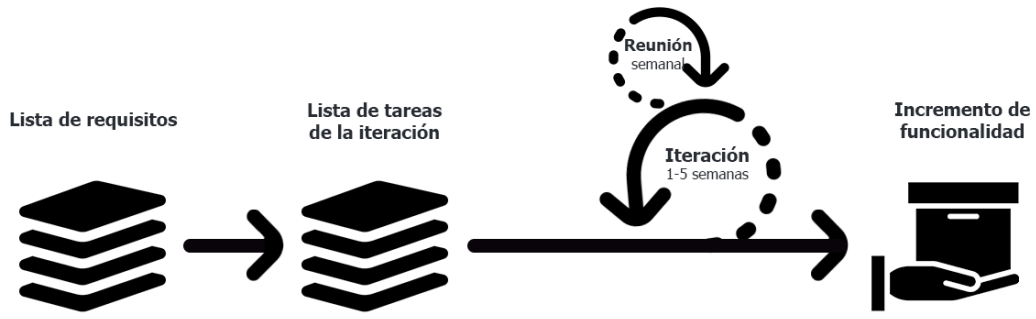


Figura 3.2: Diagrama del proceso de desarrollo adaptado.

De esta manera, se ha realizado una estimación del tiempo dedicado a cada iteración. La planificación se puede ver reflejada en la tabla 3.1.

Iteración	Descripción	Estimación
Iteración 1	Análisis de tecnologías	1 semana y 72 horas
Iteración 2	Implementación del sistema de preguntas	3 semanas y 96 horas
Iteración 3	Implementación del sistema de marcadores	3 semanas y 96 horas
Iteración 4	Implementación de la mecánica del concurso I	5 semanas
Iteración 5	Implementación de la mecánica del concuso II	5 semanas
Iteración 6	Pruebas y despliegue	2 semanas y 24 horas
Iteración 7	Documentación	3 semanas y 96 horas

Tabla 3.1: Planificación de las iteraciones

Por otro lado, las personas asignadas a los diferentes roles han sido Eusebio Angulo Sanchez-Herrera como Product Owner, Carlos González Morcillo como Scrum Master y Luis Miguel Ortiz Rozalén como Development Team.

3.2. MARCO TECNOLÓGICO

Durante el desarrollo y despliegue de *QuizzIS* se han utilizado diversas herramientas. En esta sección, se enumeran los componentes hardware y software y los lenguajes utilizados.

3.2.1. Hardware

Los elementos hardware empleados en el despliegue para la fase final de la Olimpiada han sido variados y numerosos, siendo los más importantes:

Sistema de vídeo

1 x Equipo portátil

1 x Splitter

2 x Proyectores

1 x Cable HDMI

2 x Altavoces

1 x Cable de red

Sistema de marcadores

3 x Raspberry Pi 3

3 x Pantallas

3 x Cables HDMI

3 x Pulsadores artesanales

1 x Microcontrolador Arduino

3 x Cables de red

Sistema de control

3 x Equipos portátiles

3 x Cables de red

Otros

1 x Switch

3.2.2. Software

La siguiente lista contiene los componentes software que han sido utilizados durante el desarrollo y despliegue de *QuizzIS*:

Sistema Operativo

El desarrollo del proyecto se ha llevado a cabo en un ordenador con sistema operativo **macOS** y versión Mojave.

Para el despliegue del servidor se ha utilizado el equipo mencionado anteriormente, sin embargo, para el de los clientes se utilizaron distintas distribuciones de GNU/Linux. Ha sido **Ubuntu** la distribución empleada en el sistema de preguntas y **Raspbian** en el de los marcadores.

Software de desarrollo

- **Atom:** el editor de texto que se ha empleado para la implementación.
- **Arduino IDE 1.8.9:** el entorno de desarrollo integrado de la plataforma de Arduino. Ha permitido compilar y cargar el código en la memoria del microcontrolador.
- **Google Chrome:** el navegador web que ha sido empleado para la realización de pruebas en el ordenador de desarrollo.
- **Chromium:** el navegador web que ha sido utilizado en las Raspberry Pis para el despliegue del sistema.
- **Git y GitHub:** el sistema de control de versiones distribuido y el sistema de alojamiento donde se ha almacenado el código fuente de este proyecto.

Bases de datos

Las herramientas que se han utilizado para la creación y gestión de la base de datos del *QuizzIS* han sido:

- **MongoDB Community Edition:** una base de datos NoSQL diseñada para facilitar el desarrollo y la escalabilidad. Se ha utilizado para la creación de la misma.
- **Robo 3T:** es un cliente de MongoDB que ha permitido gestionar la base de datos del sistema.

Servidor web

Para la ejecución y despliegue del servidor web se ha utilizado MAMP. Se trata de un entorno para Mac que nos permite montar de forma rápida y sencilla nuestro propio servidor local. Integra los programas software comúnmente usados para desarrollar sitios web.

Otros

Para la edición de recursos gráficos y vídeos se han utilizado:

- **Adobe Photoshop CC 2017:** un editor gráfico desarrollado por Adobe que permitió generar las imágenes que se utilizaron para la aplicación.
- **Final Cut Pro X:** una aplicación de edición desarrollada por Apple con la que se realizaron los vídeos para las preguntas del concurso.

3.2.3. Lenguajes

Por otro lado, la siguiente lista presenta os lenguajes de programación y librerías que se han utilizado para la implementación de *QuizzIS*:

- **HTML5:** un lenguaje de marcado estándar con el que se ha desarrollado la estructura y el contenido de las páginas web que conforman el *frontend* de la aplicación.
- **CSS3:** un lenguaje de diseño gráfico con el que ha sido definido y creado el estilo de las páginas web.
- **Bootstrap 4.0:** una herramienta de código abierto con la que se han hecho *responsive* las páginas web.
- **JavaScript:** un lenguaje orientado a objetos que ha sido utilizado para implementar los scripts necesarios del *frontend*.
- **JQuery:** una librería de JavaScript que ha simplificado la sintaxis y el manejo de eventos.
- **PHP 7.2:** un lenguaje de programación con el que se ha desarrollado del *backend* de la aplicación.
- **MongoDB PHP Library:** una librería de PHP que implementa una API con la que se ha gestionado la base de datos de MongoDB.
- **Python 2.7:** un lenguaje de programación interpretado de alto nivel con el que se ha realizado la comunicación con Arduino.
- **APL:** un lenguaje de programación basado en C++ con el que se han leído los eventos producidos en los pulsadores.
- **Shell:** un lenguaje de consola con el que se ha configurado el autoarranque de procesos en las Raspberry Pis.
- **LaTeX:** un lenguaje para la edición de documentos de alta calidad. Se ha utilizado para la elaboración de la documentación del proyecto.

CAPÍTULO 4

RESULTADOS

A continuación, se describe la forma en que se ha aplicado la metodología a este caso concreto y los resultados que se han obtenido.

4.1. HISTORIAS DE USUARIO

Las historias de usuario forman parte de la lista de requisitos que se realiza al inicio de cada iteración. Contienen una descripción en lenguaje común de la necesidad de un usuario al utilizar un producto. Su finalidad es la recopilación de ideas, de modo que su esfuerzo pueda ser estimado más adelante y se puedan detectar posibles problemas.

Las historias de usuario que se han extraído se enumeran a continuación:

ID	`US_VIDEOPLAYER_CONTROL`
Rol	Como técnico de control
Funcionalidad	Necesito gestionar la reproducción de contenido multimedia
Resultado	Con la finalidad de proyectar en pantalla el contenido oportuno
Aceptación	En caso de que el concurso requiera un cambio de pregunta, de fase o mostrar la clasificación, cuando el técnico utilice los controles asociados a cada acción, el sistema mostrará el recurso solicitado en la pantalla.

ID	`US_OPTION_CONTROL`
Rol	Como técnico de control
Funcionalidad	Necesito gestionar el marcado de opciones
Resultado	Con la finalidad de mostrar la opción marcada por cada equipo
Aceptación	En caso de que el equipo en posesión del turno solicite el marcado de una opción, cuando el técnico utilice uno de los controles asociados a dicha acción, el sistema mostrará la opción marcada en la pantalla.

ID	US_SCORE_CONTROL
Rol	Como técnico de control
Funcionalidad	Necesito gestionar la puntuación en los marcadores de los equipos
Resultado	Con la finalidad de mostrar la puntuación actual de cada equipo
Aceptación	En caso de que el equipo en posesión del turno acierte o falle una pregunta, cuando el técnico de control utilice uno de los controles asociados a dicha acción, el sistema actualizará la puntuación en el marcador.

ID	US_PUSHBUTTONS
Rol	Como jugador
Funcionalidad	Necesito un pulsador
Resultado	Con la finalidad de obtener el turno
Aceptación	En caso de que una ronda permita la utilización de los pulsadores, cuando uno o varios jugadores lo pulsen, el sistema gestionará una cola de turnos.

ID	US_TURN_CONTROL
Rol	Como técnico de control
Funcionalidad	Necesito gestionar el turno en los marcadores de los equipos
Resultado	Con la finalidad de indicar el equipo que tiene el turno
Aceptación	▪ En caso de que el equipo en posesión del turno acierte o falle una pregunta, cuando el técnico utilice los controles asociados a cada acción, el sistema mostrará quién ha heredado el turno en el marcador. ▪ En caso de que ningún equipo esté en posesión del turno, cuando el técnico utilice uno de los controles asociados al turno, el sistema mostrará quién ha recibido el turno en el marcador.

ID	US_SOUND_EFFECTS
Rol	Como jugador
Funcionalidad	Necesito oír efectos de sonido
Resultado	Con la finalidad de entender los eventos que ocurren
Aceptación	▪ En caso de que el concurso requiera cambiar de pregunta, marcar una opción, un acierto o un fallo, cuando el técnico utilice los controles, el sistema reproducirá el efecto de sonido correspondiente a cada evento. ▪ En caso de que una ronda permita la utilización de los pulsadores, cuando un jugador utilice el pulsador, el sistema reproducirá el efecto de sonido correspondiente a dicho evento.

ID	US_CONTEST_STATE
Rol	Como presentador
Funcionalidad	Necesito visualizar el estado del concurso
Resultado	Con la finalidad de conocer los detalles del concurso en cada momento
Aceptación	En caso de que el presentador necesite consultar el estado del concurso, cuando utilice la pestaña asociada a dicha acción, el sistema mostrará la información completa del estado del concurso.

4.2. ARQUITECTURA DEL SISTEMA

QuizzIS es un sistema distribuido con arquitectura cliente-servidor. Está formado por varios clientes que se comunican con un solo servidor a través de una red local. Cada cliente está integrado por sus propios componentes hardware y software, siendo percibidos como sistemas independientes.

Esto ha proporcionado la flexibilidad para trabajar en problemas más pequeños (Fig. 4.1):

- **Sistema de vídeo**: este sistema es el responsable de producir la salida de vídeo. A través de él, el público puede visualizar la proyección de las preguntas, los cambios de la ronda o la clasificación.
- **Sistema de marcadores:** es el sistema encargado de mostrar los nombres, puntuaciones y turnos de los equipos en los diferentes monitores. También actúa como receptor de las señales de los pulsadores.

- **Sistema de control:** este sistema está diseñado para que cada miembro del equipo de control pueda visualizar y administrar todos los detalles del concurso.
- **Servidor web:** se encuentra en uno de los portátiles de control. Él es quien recibe y responde a las solicitudes de los diferentes clientes.

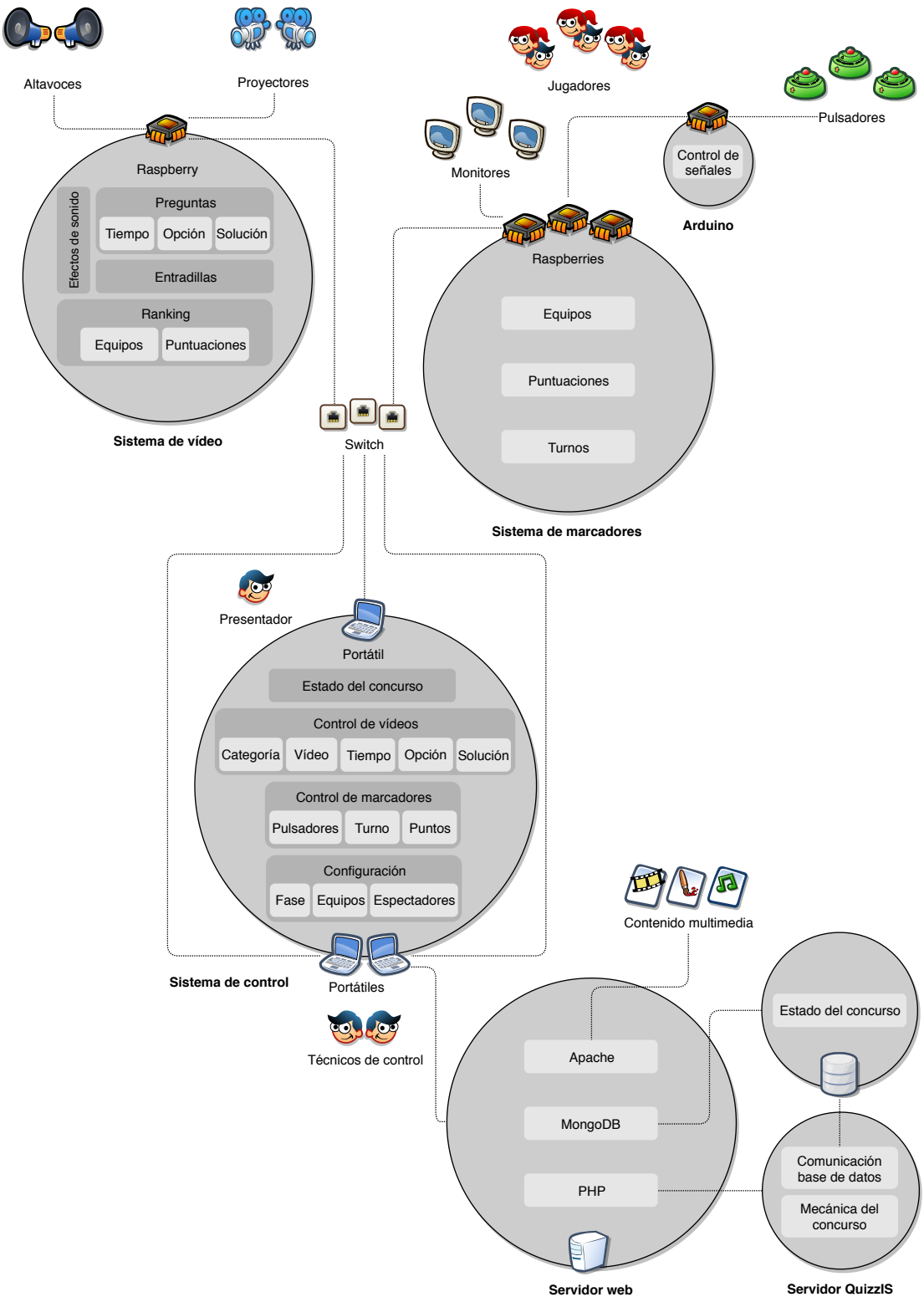


Figura 4.1: Esquema del sistema *QuizzIS*.

4.2.1. Sistema de vídeo

El sistema de vídeo consiste en una aplicación web cuyo propósito es mostrar a los jugadores y al público el desarrollo del concurso. Su tarea principal es la reproducción de contenido en formato de vídeo, ya que todas las preguntas, cambios de ronda y clasificación están codificados en MPEG-4. Sin embargo, también muestra información sobre los equipos, el cambio de ronda, la opción seleccionada y la solución. Otra de sus labores es reproducir los efectos de sonido asociados con cada evento que ocurre durante la competición.

Diseño e implementación

La implementación ha sido realizada en HTML5, JavaScript y AJAX. Esta aplicación permite recibir de forma asíncrona el estado en el que se encuentra el concurso, a fin de mostrar el contenido multimedia adecuado.

La apariencia ha sido definida con CSS3. El diseño de la aplicación se adapta completamente a la pantalla, de modo que cuando se proyecta, no se observa ningún otro elemento que no forme parte del vídeo que se está reproduciendo.

El funcionamiento de la aplicación es posible gracias a AJAX. Con esta técnica es posible realizar un *polling* con el que se consulta constantemente el estado del concurso al servidor. De esta manera, el contenido se muestra automáticamente a medida que cambia.

La función *runPoll* (Listado 4.1) materializa este comportamiento. Consiste básicamente en una llamada recursiva en intervalos de tiempo. En cada llamada, se realiza una solicitud HTTP al servidor y, si obtiene respuesta, actualiza el contenido de la aplicación con el estado del concurso.

Listado 4.1: Código fuente de la función *runPoll*.

```
function runPoll() {
  setTimeout(function() {
    $.ajax({
      url: 'http://192.168.0.8:8888/server/server.php',
      success: function(response) {
        checkContestState(response);
      },
      dataType: 'json',
      complete: function() {
        runPoll();
      }
    });
  }, 100);
}
```

Las peticiones HTTP se producen de forma asíncrona, es decir, no requieren actualizar toda la página. Esto sucede mediante un objeto de tipo XMLHttpRequest que el cliente crea en cada solicitud. A través de este objeto se establece una comunicación en segundo plano con el servidor (Fig. 4.2).

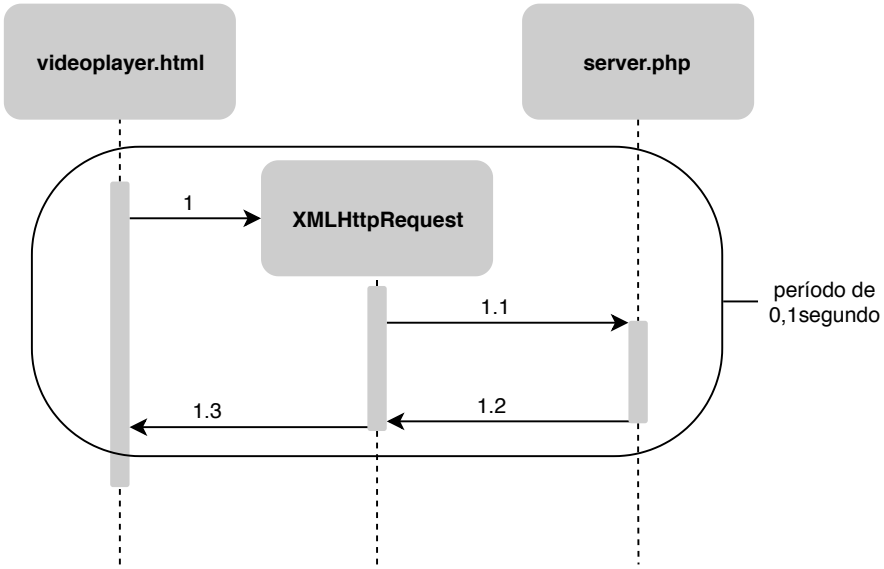


Figura 4.2: Diagrama de secuencia del *polling*.

Una vez que se completa la solicitud, se recibe un objeto *response* como respuesta, que es manipulado para mostrar su contenido en diferentes secciones de la página web. Este objeto es de tipo JSON y está formado por una colección de pares clave/valor (Fig. 4.3).


```
▼ [{_id: "phase", value: "bach"}, {_id: "mode", value: "leads"}, {_id: "lead", value: "0"},…]
  ▶ 0: {_id: "phase", value: "bach"}
  ▶ 1: {_id: "mode", value: "leads"}
  ▶ 2: {_id: "lead", value: "0"}
  ▶ 3: {_id: "category", value: "mandatory"}
  ▶ 4: {_id: "mandatoryBach", value: [0, 10]}
  ▶ 5: {_id: "generalBach", value: [0, 32]}
  ▶ 6: {_id: "mandatoryCycl", value: [0, 10]}
  ▶ 7: {_id: "generalCycl", value: [0, 41]}
  ▶ 8: {_id: "play", value: true}
  ▶ 9: {_id: "timer", value: "-1"}
  ▶ 10: {_id: "option", value: 0}
  ▶ 11: {_id: "solution", value: false}
  ▶ 12: {_id: "lock", value: true}
  ▶ 13: {_id: "turn", value: [null, null, null], buffer: ""}
  ▶ 14: {_id: "teamBachA", value: ["StackOverF", 5]}
  ▶ 15: {_id: "teamBachB", value: ["Buscaminas", 10]}
  ▶ 16: {_id: "teamBachC", value: ["Revienta R.", 15]}
  ▶ 17: {_id: "teamCyclA", value: ["Array Of Null", 5]}
  ▶ 18: {_id: "teamCyclB", value: ["404TNNF", 10]}
  ▶ 19: {_id: "teamCyclC", value: ["EstasOverF", 15]}
  ▶ 20: {_id: "sfx", value: ""}
  ▶ 21: {_id: "spectators", value: ["", "", ""]}
```


Figura 4.3: Respuesta del servidor en formato JSON.

A partir de este objeto, se toman los pares que contienen la información que requiere cada modo de vídeo: **pregunta**, **cambio de ronda** o **clasificación**.

La Figura 4.4 es una representación tridimensional el reproductor de vídeo en modo pregunta. En la parte trasera se encuentra el vídeo que muestra la pregunta y las respuestas. En superposición, se agrega la información extraída del estado del concurso. Así, en la parte superior se ubican los equipos con sus respectivos puntajes. El tiempo restante se encuentra en la esquina superior derecha. Y finalmente, la opción, que se superpone a la respuesta que ha sido seleccionada por el equipo. Si es necesario mostrar la solución, el vídeo se oculta y en su lugar se muestra una imagen que contiene la solución correcta.

Figura 4.4: Representación 3D del reproductor en modo pregunta.

Por lo tanto, el reproductor en modo pregunta debe conocer:

- ***phase***: es la fase del concurso destinada a cada nivel de estudios. Por lo tanto, puede tomar uno de los siguientes valores: *bach* o *cycl*.
- ***mode***: es el modo de vídeo del reproductor. En este caso, el valor asignado es *questions*.
- ***category***: es la categoría a la que pertenece la pregunta actual. Puede ser uno de estos dos valores: *mandatory* o *general*.
- ***mandatoryBach***: es un vector de dos posiciones. En la primera, se almacena el número asociado con la última pregunta obligatoria realizada a los estudiantes de

bachillerato. En la segunda posición, el número total de preguntas obligatorias de bachillerato.

- ***generalBach***: es un vector de dos posiciones. En la primera, se almacena el número asociado con la última pregunta de cultura general realizada a los estudiantes de bachillerato. En la segunda posición, el número total de preguntas de cultura general de bachillerato.
- ***mandatoryCycl***: es un vector de dos posiciones. En la primera, se almacena el número asociado con la última pregunta obligatoria realizada a los estudiantes de ciclos formativos. En la segunda posición, el número total de preguntas obligatorias de ciclos formativos.
- ***generalCycl***: es un vector de dos posiciones. En la primera, se almacena el número asociado con la última pregunta de cultura general realizada a los estudiantes de ciclos formativos. En la segunda posición, el número total de preguntas de cultura general de ciclos formativos.
- ***play***: es un valor binario que indica si el vídeo debe reproducirse o pausarse.
- ***timer***: es un valor entero que indica el tiempo restante de la ronda actual.
- ***option***: es un código de dos dígitos asociado con cada opción que el equipo puede marcar.
- ***solution***: es un valor que indica si se debe mostrar o no la solución.
- ***teamBachA***: contiene un vector con el nombre y la puntuación del primer equipo de bachillerato.
- ***teamBachB***: contiene un vector con el nombre y la puntuación del segundo equipo de bachillerato.
- ***teamBachC***: contiene un vector con el nombre y la puntuación del tercer equipo de bachillerato.
- ***teamCyclA***: contiene un vector con el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclB***: contiene un vector con el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclC***: contiene un vector con el nombre y la puntuación del primer equipo de ciclos formativos.
- ***sfx***: es un valor numérico asociado a cada efecto de sonido que se produce.

La Figura 4.5 muestra la interfaz del reproductor de vídeo en modo clasificación. En la parte de detrás se sitúa el vídeo que contiene el podio. De forma superpuesta, se añade el resto de la información. Así, en la parte superior se ubica la fase actual del concurso y, encima de cada escalón del podio, se encuentran los equipos con sus respectivos puntos.

Figura 4.5: Captura de la interfaz del reproductor en modo clasificación.

En consecuencia, el reproductor en modo clasificación debe saber:

- ***phase***: *bach* o *cycl*, dependiendo de la fase actual.
- ***mode***: en este caso, el valor asignado es *ranking*.
- ***teamBachA***: el nombre y la puntuación del primer equipo de bachillerato.
- ***teamBachB***: el nombre y la puntuación del segundo equipo de bachillerato.
- ***teamBachC***: el nombre y la puntuación del tercer equipo de bachillerato.
- ***teamCyclA***: el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclB***: el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclC***: el nombre y la puntuación del primer equipo de ciclos formativos.

La Figura 4.6 muestra la interfaz del reproductor de video en modo cambio de ronda. Esta vez la aplicación solo deja visible el elemento de vídeo.

Figura 4.6: Captura de la interfaz del reproductor en modo cambio de ronda.

Por ello, el reproductor en modo cambio de ronda solo debe conocer:

- ***mode***: en este caso, el valor asignado es *leads*.
- ***leads***: el identificador que corresponde a cada cambio de ronda.

Despliegue

El despliegue de esta aplicación se ha llevado a cabo en una Raspberry Pi (Fig. 4.7). Este ordeandor está configurado para ejecutar automáticamente un cliente web a pantalla completa. Por medio de un archivo de configuración contenido en una memoria USB, se indica la forma en que debe iniciarse, en este caso, *videoplayer*.

Los dispositivos de salida que se conectan a esta Raspberry son:

- Una memoria USB de donde se lee el rol que debe adquirir la Raspberry.
- Un *splitter*, que distribuye el mismo contenido a dos proyectores orientados hacia las paredes
- Un cable de red responsable de la comunicación con el resto del sistema.
- Unos altavoces, necesarios para transmitir el sonido generado por la aplicación web.

(a) Conexiones del sistema de vídeo

(b) Proyección de la aplicación web de vídeo

Figura 4.7: Despliegue del sistema de vídeo.

4.2.2. Sistema de marcadores

El sistema de marcadores consiste en una aplicación web más simple, cuyo propósito es presentar al público la información de cada equipo. Sin embargo, también tiene un módulo para capturar el orden en que se solicitan los turnos a través de los pulsadores. Por lo tanto, es responsable de mostrar los nombres, las puntuaciones y los turnos correspondientes.

Diseño e implementación

La implementación de la aplicación también se ha llevado a cabo en HTML5, JavaScript y AJAX. Esta aplicación recibe los datos relacionados con el equipo a través del *polling* (Listado 4.1). De esta manera, el contenido se actualiza a medida que cambia.

Asimismo, la apariencia se ha definido con CSS3 y el diseño de la aplicación se adapta completamente a la pantalla.

Si bien, este sistema utiliza un microcontrolador adicional: una placa Arduino. El Arduino se encarga de digitalizar los eventos que los jugadores producen en los pulsadores. Su circuito integrado interactúa con el programa que se carga en la placa.

El programa está implementado en C++. Se compone de dos funciones principales: *setup* y *loop*. En la función *setup* se inicializan los pines, se configura el puerto serie y comienza la transmisión de datos. El código de la función *loop* se ejecuta en un bucle. En cada iteración se verifica si se ha presionado uno de los botones y, si es así, se envía el identificador del equipo a través del puerto serie (Listado 4.2).

Listado 4.2: Código fuente cargado en Arduino.

```
#define BUTTONS 3
int buttons[] = {2, 3, 4}; // Pulsadores
int pressed[] = {0, 0, 0}; // Presionado
int LEDs[] = {10, 11, 12}; // LEDs
char events[] = {'a', 'b', 'c'}; // Eventos

void setup() {
  for (int i = 0; i < BUTTONS; i++) {
    pinMode(buttons[i], INPUT); // Pulsadores
    pinMode(LEDs[i], OUTPUT); // LEDs
  }
  Serial.begin(9600); // Puerto serie
}

void setPressed(int i) {
  if (pressed[i])
    return;
  digitalWrite(LEDs[i], HIGH);
  pressed[i] = 1;
  Serial.print(events[i]);
}

void clearPressed(int i) {
  if (!pressed[i])
    return;
  digitalWrite(LEDs[i], LOW);
```

```
27   pressed[i] = 0;
28 }
29
30 void loop() {
31   for (int i = 0; i < BUTTONS; i++)
32   if (digitalRead(buttons[i]) == HIGH)
33     setPressed(i);
34   else
35     clearPressed(i);
36   delay(10);
37 }
```

A fin de permitir la comunicación entre la Raspberry y el Arduino, se ha programado un script de Python que lee los datos del puerto serie. En este script, se inicializa el puerto serie y se crea un hilo que ejecuta una función en la que se concatenan las señales que van siendo leídas. La cadena de caracteres resultante se envía al servidor. (Listado 4.3).

Listado 4.3: Script para la lectura de eventos de los pulsadores.

```
1  from serial import *
2  from threading import Thread
3  import requests
4
5  last_received = ''
6
7  def receiving(ser):
8    global last_received
9
10   buffer_string = ''
11   while True:
12     buffer_string = buffer_string + ser.read(ser.inWaiting())
13     if '\n' in buffer_string:
14       lines = buffer_string.split('\n')
15       last_received = lines[-2]
16       buffer_string = lines[-1]
17     requests.post(url = ↩
         ↪ 'http://192.168.0.8:8888/server/server.php', ↩
         ↪ data = {'turn' : buffer_string})
18
19 if __name__ ==  '__main__':
20   ser = Serial(
21     port='/dev/ttyACM0',
22     baudrate=9600,
23     bytesize=EIGHTBITS,
24     parity=PARITY_NONE,
25     stopbits=STOPBITS_ONE,
26     timeout=0.1,
```

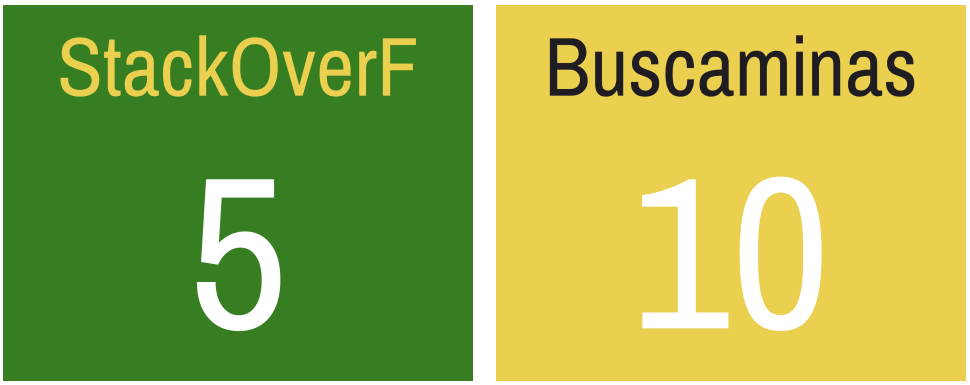
```
27      xonxoff=0,
28      rtscts=0,
29      interCharTimeout=None
30  )
31  Thread(target=receiving, args=(ser,)).start()
```

Del objeto *response* (Fig. 4.3) se toman los pares que contienen la información que requiere el marcador.

La Figura 4.8 muestra la interfaz del marcador. Dispone de un solo contenedor que incluye una etiqueta con el nombre del equipo y otra con su puntuación.

Figura 4.8: Captura de la interfaz del marcador.

Cuando un equipo adquiere el primer turno, el color de fondo del contenedor cambia a verde. Si alguno de los otros equipos ha solicitado el turno, el fondo se muestra en amarillo.



(a) Colores para el primer turno (b) Colores para el resto de turnos

Figura 4.9: Representación cromática de los turnos.

Por lo tanto, el marcador debe conocer:

- ***lock***: es un valor que indica si los equipos pueden solicitar el turno o no.
- ***turn***: es un vector que contiene el orden de los turnos.
- ***teamBachA***: el nombre y la puntuación del primer equipo de bachillerato.
- ***teamBachB***: el nombre y la puntuación del segundo equipo de bachillerato.
- ***teamBachC***: el nombre y la puntuación del tercer equipo de bachillerato.
- ***teamCyclA***: el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclB***: el nombre y la puntuación del primer equipo de ciclos formativos.
- ***teamCyclC***: el nombre y la puntuación del primer equipo de ciclos formativos.

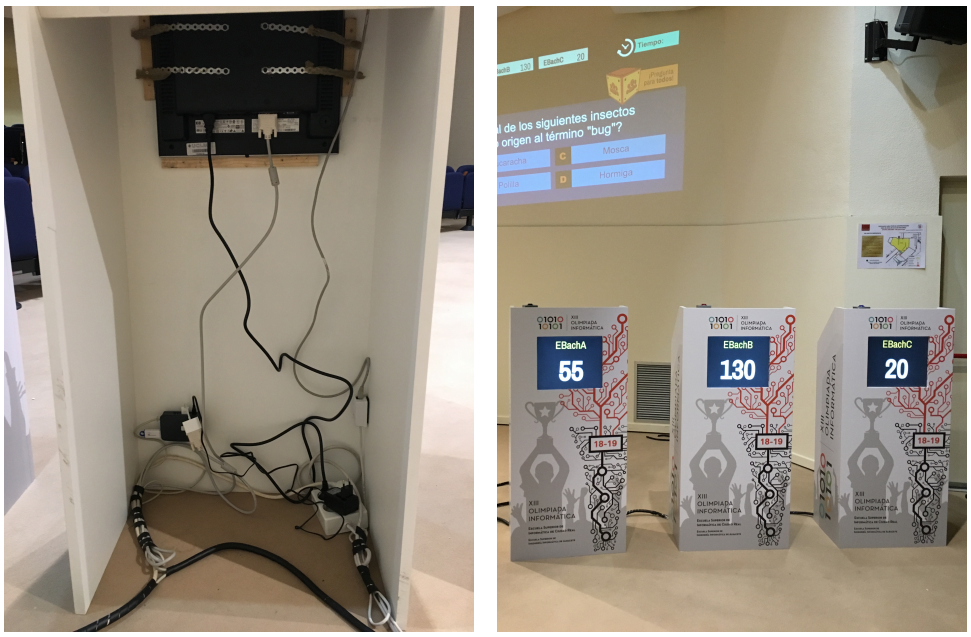
Despliegue

El despliegue de esta aplicación ha sido llevado a cabo en tres Raspberry Pi (Fig. 4.10). También se han configurado para iniciar automáticamente un cliente web a pantalla completa. En el archivo de configuración de la memoria extraíble se informa de que debe iniciarse en modo *videomarker*, acompañado del equipo que va a representar.

Los dispositivos de salida que se conectan a estos ordenadores son:

- Una memoria USB de donde se lee el rol que debe adquirir la Raspberry.
- Un monitor a través del que se muestra la interfaz de la aplicación.
- Un cable de red responsable de la comunicación con el resto del sistema.

- Un Arduino que captura los eventos producidos sobre los pulsadores.



(a) Conexiones del sistema de marcadores (b) Visualización de la aplicación web de marcadores

Figura 4.10: Despliegue del sistema de marcadores.

4.2.3. Sistema de control

4.2.4. Servidor web

4.3. DISTRIBUCIÓN DEL TRABAJO

4.3.1. Iteración 1: Análisis de tecnologías

4.3.2. Iteración 2: Implementación del sistema de preguntas

4.3.3. Iteración 3: Implementación del sistema de marcadores

4.3.4. Iteración 4: Implementación de la mecánica del concurso I

4.3.5. Iteración 5: Implementación de la mecánica del concurso II

4.3.6. Iteración 6: Pruebas y despliegue

4.3.7. Iteración 7: Documentación

4.4. ESTADÍSTICAS DEL REPOSITORIO

Figura 4.11: Relación de lenguajes generada por *GitHub*.

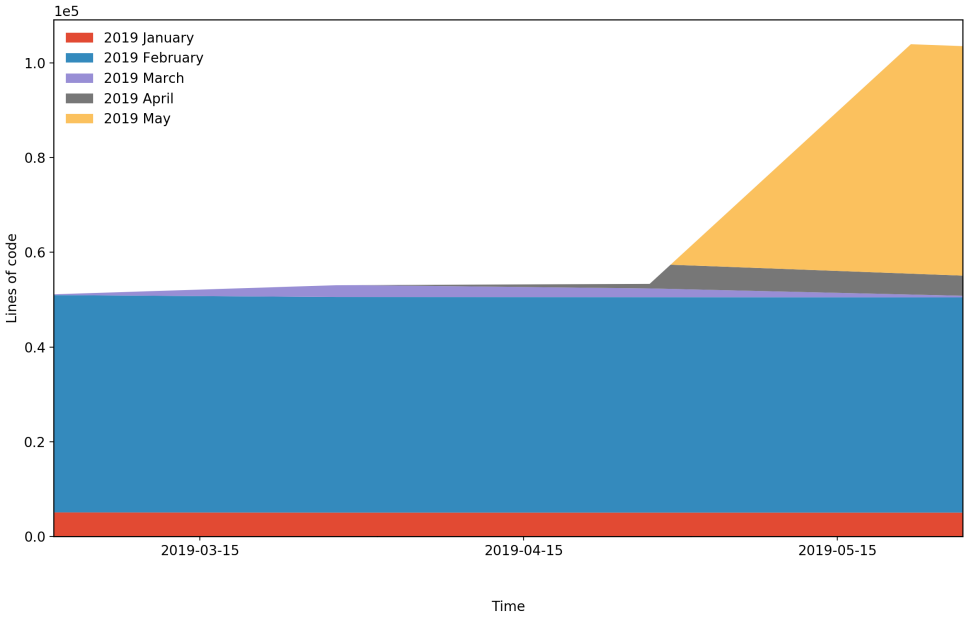


Figura 4.12: Diagrama *burn down* del repositorio obtenido con *hercules*.

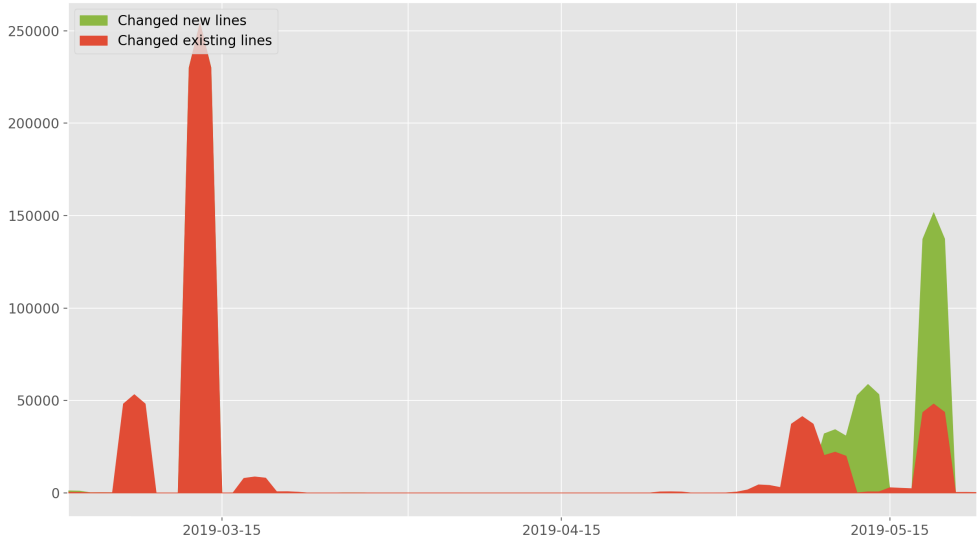


Figura 4.13: Diagrama de líneas agregadas o cambiadas obtenido con *hercules*.

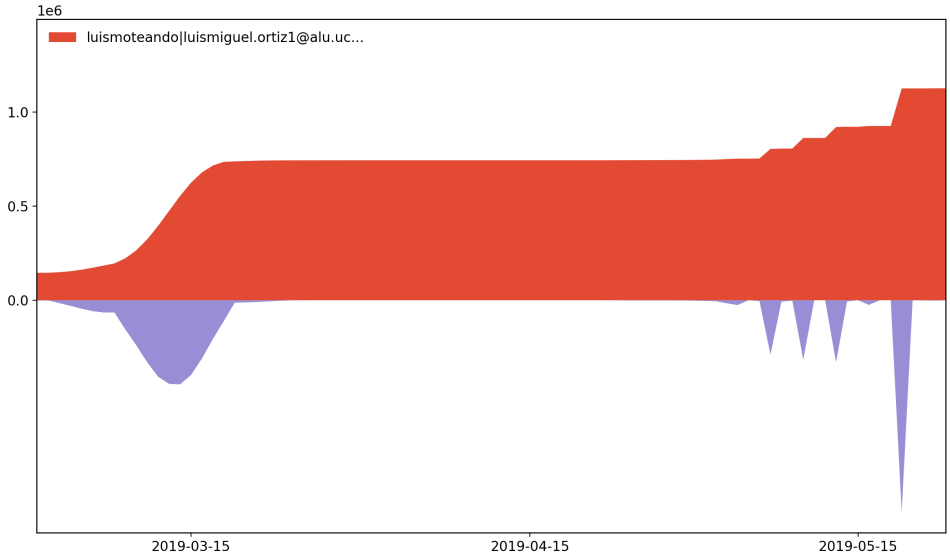


Figura 4.14: Diagrama de esfuerzos obtenido con *hercules*.

CAPÍTULO 5

CONCLUSIONES

En este capítulo se realizará un juicio crítico y discusión sobre los resultados obtenidos. Si es pertinente deberá incluir información sobre trabajos derivados como publicaciones o ponencias, así como trabajos futuros, solo si estos están planificados en el momento en que se redacta el texto. Además incluirá obligatoriamente la explicación de cómo el trabajo realizado satisface las competencias de la tecnología específica cursada.

ANEXO A

EL PRIMER ANEXO

En los anexos se incluirá de modo opcional material suplementario que podrá consistir en breves manuales, listados de código fuente, esquemas, planos, etc. Se recomienda que no sean excesivamente voluminosos, aunque su extensión no estará sometida a regulación por afectar esta únicamente al texto principal.

Bibliografía Esta sección, que si se prefiere puede titularse «Referencias», incluirá un listado por orden alfabético (primer apellido del primer autor) con todas las obras en que se ha basado para la realización del TFG en las que se especificará: autor/es, título, editorial y año de publicación. Solo se incluirán en esta sección las referencias bibliográficas que hayan sido citadas en el documento. Todas las fuentes consultadas no citadas en el documento deberían incluirse en una sección opcional denominada «Material de consulta», aunque preferiblemente estas deberían incluirse como referencias en notas a pie de página a lo largo del documento.

Se usará método de citación numérico con el número de la referencia empleada entre corchetes. La cita podrá incluir el número de página concreto de la referencia que desea citarse. Debe tenerse en cuenta que el uso correcto de la citación implica que debe quedar claro para el lector cuál es el texto, material o idea citado. Las obras referenciadas sin mención explícita o implícita al material concreto citado deberían considerarse material de consulta y por tanto ser agrupados como «Material de consulta» distinguiéndolas claramente de aquellas otras en las que si se recurre a la citación.

Cuando se desee incluir referencias a páginas genéricas de la Web sin mención expresa a un artículo con título y autor definido, dichas referencias podrán hacerse como notas al pie de página o como un apartado dedicado a las «Direcciones de Internet».

Todo el material ajeno deberá ser citado convenientemente sin contravenir los términos de las licencias de uso y distribución de dicho material. Esto se extiende al uso de diagramas y fotografías. El incumplimiento de la legislación vigente en materia de protección de la propiedad intelectual es responsabilidad exclusiva del autor del trabajo independientemente de la cesión de derechos que este haya convenido. De este modo será responsable legal ante

cualquier acción judicial derivada del incumplimiento de los preceptos aplicables. Así mismo ante dicha circunstancia los órganos académicos se reservan el derecho a imponer al autor la sanción administrativa que se estime pertinente.

Índice temático Este índice es opcional y se empleará como índice para encontrar los temas tratados en el trabajo. Se organizará de modo alfabético indicando el número de página(s) en el que se aborda el tema concreto señalado.

REFERENCIAS

[Bec+01] Kent Beck y col. *Principios del Manifiesto Ágil*. 2001. URL: http://agilemanifesto.org/iso/es/principles.html (visitado 04-06-2019).

[Cad+13] Andrés Navarro Cadavid y col. «Revisión de Metodologías Ágiles para el Desarrollo de Software». En: *Prospectiva* 11.2 (2013), págs. 30-39.

[CIT+18] CITIPA y col. *Olimpiada de Ingeniería Informática de Asturias*. 26 de abr. de 2018. URL: http://impulsotic.org/olimpiadainformatica/ (visitado 15-04-2019).

[CL12] José H. Canós y Mª Carmen Penadés Patricio Letelier. «Metodologías Ágiles en el Desarrollo de Software». En: (2012), pág. 1.

[Din+12] Torgeir Dingsøyr y col. «A Decade of Agile Methodologies: Towards Explaining Agile Software Development». En: *Journal of Systems and Software* 85.6 (2012), págs. 1213-1221. URL: http://www.sciencedirect.com/science/article/pii/S0164121212000532.

[Let13] Patricio Letelier. *Agility at Work: ¿Kanban O Scrum?: That Is Not the Question*. 4 de mayo de 2013. URL: http://agilismoatwork.blogspot.com/2013/05/kanban-o-scrum-esta-no-es-la-cuestion.html (visitado 05-06-2019).

[Mar13] Eduardo Monroy Martínez. «QuQuSI: Plataforma para la Gestión Integral de Concursos Televisivos». En: (2013), págs. 1-11.

[Sch04] Ken Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[SS10] Ken Schwaber y Jeff Sutherland. *What Is Scrum?* 2010. URL: http://www.volaroint.com/wp-content/uploads/dlm_uploads/2014/03/DC-VOLARO-Training-Scrum-What_Is_Scrum.pdf (visitado 04-06-2019).