Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO
INDUSTRIA INGENIARITZA TEKNIKORAKO ATALA
SECCIÓN INGENIERÍA TÉCNICA INDUSTRIAL

Grado en Ingeniería Informática de Gestión y Sistemas de Información

TRABAJO FIN DE GRADO

2016 / 2017

KuroObi

Memoria

DATOS DE LA ALUMNA O DEL ALUMNO	DATOS DEL DIRECTOR O DE LA DIRECTORA
NOMBRE Jonatan	NOMBRE JUAN ANTONIO
APELLIDOS Guijarro Garcia	APELLIDOS PEREIRA VARELA
	DEPARTAMENTO Lenguajes y Sistemas Informáticos
FDO.:	FDO.:
FECHA: 22-10-2016	FECHA: 22-10-2016

Anexo II

RESUMEN

El proyecto se basa en una página web para gestionar las clases de karate desde la perspectiva de un profesor y para auto-gestionarse desde la vista de un alumno.

El profesor dispone de diferentes herramientas para gestionar a sus alumnos, como pueden ser, rellenar fichas sacando una foto, pasar lista deslizando el dedo sobre la foto del alumno o rellenar un examen entre otros.

El alumno puede grabar los entrenamientos y/o la realización de diferentes ejercicios para subirlos a diferentes sistemas de almacenamiento online y compartirlo con el profesor.

El proyecto está realizado en su mayoría en PHP y JavaScript. Se usa el framework *CodeIgniter* para agilizar ciertas operaciones, sobre todo la conexión entre el modelo, la vista y el controlador.

Se incluyen diferentes *API*s para dotar al proyecto de funcionalidades más completas, las *API*s usadas son: YouTube, Google Drive y VanillaForum.

Los datos se guardan en una base de datos MySQL para poder ser almacenados y consultados de manera rápida y sencilla.

Tabla de contenido

1. Introducción 1
1.1 Origen del proyecto 1
1.2 Razones de elección del TFG 2
1.3 Planteamiento del problema 2
1.4 Glosario de términos 3
2. Planteamiento inicial 5
2.1 Objetivos del proyecto 5
2.2 Alcance del proyecto 6
2.2.1 Fases del proyecto 6
2.2.2 Estructura de Descomposición del Trabajo (EDT) 7
2.2.3 Tareas 8
2.3 Planificación temporal 26
2.4 Arquitectura 29
2.4.1 Arquitectura interna 29
2.4.2 Arquitectura externa 30
2.4.3 Arquitectura cliente-servidor 30
2.5 Herramientas 31
2.5.1 Hardware: 31
2.5.2 Software: 31
2.5.3 Lenguajes de programación 32
2.5.4 Frameworks 33
2.6 Gestión de riesgos 33
2.6.1 Planificación Incorrecta del tiempo del proyecto. 34
2.6.2 Baja por accidente o enfermedad 34
2.6.3 Pérdidas totales o parciales de documentos/ficheros 34
2.6.4 Sufrir fallos en el ordenador de trabajo. 35
2.6.5 Problemas a la hora de implementar algoritmo 35
2.6.6 Problemas a la hora de implementar prototipos debido a cambios en las *API*. 35
2.6.7 Problemas personales. 36
2.7 Evaluación económica 36
2.7.1 Recuperar la inversión 37
3. Antecedentes 39
3.1 Historia 39

3.2 Situación actual 39
3.3 Estudio de diferentes alternativas. 39
4. Captura de requisitos 41
4.1 Requisitos previos 41
4.2 Casos de uso 42
4.2.1 Jerarquía de actores 42
4.2.2 Modelo de casos de uso 43
4.3 Modelo de dominio 46
5. Análisis y diseño 49
5.1 Diseño de la base de datos 49
5.1.1 Modelo relacional y descripción detallada de la base de datos. 50
5.2 Estructura de la aplicación web 54
5.2.1 Separación modelo-vista-controlador 54
5.2.1.1 Modelo 56
5.2.1.2 Vista 57
5.2.1.3 Controlador 58
5.2.2 *API* 59
5.2.2.1 *API* de *YouTube* 59
5.2.2.2 *API de Google Drive* 59
5.2.2.3 *API* de *VanillaForum* 60
5.2.2.4 *API* de *Google Chart* 60
6. Desarrollo 61
6.1 Instalación previa. 61
6.2 Diseño de la web 61
6.2.1 Página principal 62
6.2.2 Registro de nuevos usuarios 62
6.2.3 Página usuario profesor 63
6.2.4 Página usuario alumno 70
6.3 Diseño del código 73
6.3.1 Configuración de *CodeIgniter* 73
6.3.2 Tabla de alumnos 74
6.3.3 Dar de alta nuevo alumno 74
6.3.4 Asistencia 76
6.3.5 Examen 76
6.3.6 *API* de *YouTube* 78

6.3.7 *API de Google Drive* 80
6.3.8 *API* de *VanillaForum* 81
6.4 Responsive 81
7 Verificación y pruebas 85
7.1 Objetivo 85
7.2 Registrar usuario 85
7.3 Login 86
7.4 Nuevo alumno 86
7.5 Ver alumno 86
7.6 Editar alumno 87
7.7 Gestión de alumnos 87
7.8 Control de asistencia 88
7.9 Tabla de asistencia 88
7.10 Tabla examen 89
7.11 Examen 89
7.12 Ver examen 89
7.12 Añadir entrenamientos 90
7.13 Tabla de entrenamientos 90
7.14 Ver entrenamientos. 91
7.15 Foro 91
7.16 Perfil profesor 92
7.17 Logros y progresión 92
7.18 Grabar sesión alumno 92
7.19 Ver sesión 93
7.20 Notificar sesión 93
7.21 Revisar notificación 93
7.22 Diseño 93
7.23 Seguridad 94
8 LOPD 95
9 Conclusiones 97
9.1 Planificación inicial Vs planificación real. 97
9.2 Futuro para KuroObi 100
9.3 Reflexión personal 100
10 Agradecimientos 101
11 Bibliografía 102

Anexo I – Casos de uso extendidos 104
Anexo II - Diagramas de secuencia 114

Índice de ilustraciones

Ilustración 1: EDT 7
Ilustración 2 Gantt 1 28
Ilustración 3 Gantt II 28
Ilustración 4 Arquitectura interna de la web 29
Ilustración 5 Arquitectura externa de la web 30
Ilustración 6 Cliente/servidor[2] 31
Ilustración 7 Sublime Text 3 31
Ilustración 8 Google Chrome 31
Ilustración 9 Wamp Server 32
Ilustración 10 Microsoft Office 32
Ilustración 11 SourceTree 32
Ilustración 12 Visual Paradigm 32
Ilustración 13 PHP 32
Ilustración 14 Bootstrap 33
Ilustración 15 CodeIgniter 33
Ilustración 16 Jerarquía de actores 43
Ilustración 17 C.U. Usuario no-identificado 43
Ilustración 18 C.U. U. Profesor 44
Ilustración 19 C.U. U. Alumno 44
Ilustración 20 Modelo de dominio 47
Ilustración 21 Esquema Bd 49
Ilustración 22Interface phpMyAdmin 50
Ilustración 23 Kt_users 50
Ilustración 24 Tabla asistencia 51
Ilustración 25 Tabla examen 52
Ilustración 26 Esquema BBDD 53
Ilustración 27 CodeIgniter 1 55
Ilustración 28 Carpeta applications 56
Ilustración 29 Carpeta Models 56
Ilustración 30 Profesor_model 57
Ilustración 31 Carpeta views 57
Ilustración 32 Carpeta controllers 58
Ilustración 33 Archivo controlador 58
Ilustración 34 Menú 62
Ilustración 35 Registro 62
Ilustración 36 Notificaciones 63
Ilustración 37 Gestionar alumnos 64
Ilustración 38 Dar de alta alumno 65
Ilustración 39 Gestión de alumnos 65
Ilustración 40 Tabla examen 66
Ilustración 41 Pantalla examen 66
Ilustración 42 Gestionar entrenamiento 68
Ilustración 43 Foro 68
Ilustración 44 Borrado de exámenes 69

Ilustración 45 Lista de alumnos 69
Ilustración 46 Progresiones y logros 70
Ilustración 47 Gestionar sesiones 71
Ilustración 48 Ver sesión 71
Ilustración 49 Crear waza 72
Ilustración 50 Ver waza 72
Ilustración 51 Ejemplo config 73
Ilustración 52 Ejemplo configuración 73
Ilustración 53 Tabla filtrable 74
Ilustración 54 Capturar imagen 74
Ilustración 55 Controlador users 75
Ilustración 56 Conexión modelo 75
Ilustración 57 Modelo registro 75
Ilustración 58 Swipe 76
Ilustración 59 Círculo de puntuación 77
Ilustración 60 Conexión JavaScript 77
Ilustración 61 Credenciales YouTube 78
Ilustración 62 Capturar vídeo 79
Ilustración 63 Api YouTube 79
Ilustración 64 Enlace API YouTube 80
Ilustración 65 Código Google Drive 80
Ilustración 66 API Google Drive 80
Ilustración 67 Script VanillaForum 81
Ilustración 68 Responsive ej.1 82
Ilustración 69 Responsive ej.2 82
Ilustración 70 Responsive ej.3 83
Ilustración 71 Responsive ej.4 83
Ilustración 72 Planificación inicial Vs final 98
Ilustración 73 Gantt final 99
Ilustración 74 Gantt final 2 99
Ilustración 75 Gantt final 3 99
Ilustración 76 Asistencia alumno ausente 105
Ilustración 77 Asistencia alumno presente 105
Ilustración 78 Iniciar examen 106
Ilustración 79 Tabla examen 106
Ilustración 80 Interface examen I 107
Ilustración 81 Interface examen II 107
Ilustración 82 Comentario examen 108
Ilustración 83 Selección de archivo 108
Ilustración 84 Progreso vídeo 108
Ilustración 85 Exámenes realizados 108
Ilustración 86 Gestionar wazas 109
Ilustración 87 Elegir waza 110
Ilustración 88 Crear wazas fotos 110
Ilustración 89 Mensaje crear wazas 111
Ilustración 90 Ver waza 111
Ilustración 91 Icono principal waza 111

Ilustración 92 D.S Control asistencia 114
Ilustración 93 D.S. Realizar examen 118
Ilustración 94 D. S Grabar sesiones..... 122
Ilustración 95 D.S Crear waza..... 123

Índice de tablas

Tabla 1 Estimación temporal 27
Tabla 2 Tabla de gastos 37
Tabla 3 Registrar usuario 85
Tabla 4 Login 86
Tabla 5 Nuevo alumno 86
Tabla 6 Ver alumno 86
Tabla 7 Editar alumno 87
Tabla 8 Gestión alumnos 87
Tabla 9 Control asistencia 88
Tabla 10 Tabla asistencia 88
Tabla 11 Tabla examen 89
Tabla 12 Examen 89
Tabla 13 Ver examen 89
Tabla 14 Añadir entrenamientos 90
Tabla 15 Tabla de entrenamientos 90
Tabla 16 Ver entrenamientos 91
Tabla 17 Foro 91
Tabla 18 Perfil profesor 92
Tabla 19 Logros y progresión 92
Tabla 20 Grabar sesión alumno 92
Tabla 21 Ver sesión 93
Tabla 22 Notificar sesión 93
Tabla 23 Revisar notificación 93
Tabla 24 Diseño 94
Tabla 25 Seguridad 94

1. Introducción

A lo largo de este documento se mostrará el proceso de desarrollo del Trabajo Fin de Grado (a partir de ahora TFG) **"KuroObi"** (cinturón negro en japonés)**.** Este capítulo será el punto de entrada a la documentación, donde se mostrará una visión general sobre el tema, las motivaciones que llevaron a su creación y las tecnologías que se usarán.

1.1 Origen del proyecto

Hoy en día la tecnología llega a cada rincón del planeta, es un proceso ligado a la evolución tecnológica que busca abarcar todos los ámbitos posibles. Hemos sido testigos de cómo un móvil se ha convertido en nuestro centro de noticias, de cultura, de humor y de apertura social. El concepto *Internet of Things (IoT o internet de las cosas)* que hace referencia a la interconexión digital que hay entre los objetos cotidianos, ha pasado de ser un concepto poco conocido a estar presente en muchas de las noticias tecnológicas que escuchamos y vemos hoy en día. Por ejemplo, ya hay coches con *Wi-Fi* en su interior para favorecer la interconexión tanto del coche como de los ocupantes a través de dispositivos móviles u ordenadores portátiles.

Pero hay entornos donde esta entrada de la tecnología digital es más limitada. Quizá porque aportan poco a esta evolución tecnológica o porque están cerrados a esta evolución.

El caso que nos ocupa en este TFG quizá sea más lo segundo. El karate tiene ese halo de tradicionalismo y antigüedad que no invita a la inserción tecnológica más lejos de mejores guantillas y protecciones o alguna cámara de vídeo para grabar sesiones. Y quizá sea mejor mantener ese ambiente, para mantener esa similitud con los creadores del karate tradicional que solo disponían de un *Dojo* (lugar donde se practica el karate) y de sus compañeros para perfeccionar y mejorar su karate sin disponer de nada más.

Pero hay aspectos no estrictamente relacionados con el karate que sí permiten mejoras para ser más accesibles, cómodos y facilitar el desempeño de, en este caso, el profesor de karate y el alumno. Son aspectos más enfocados a la gestión de las clases que al contenido de las mismas.

Tareas como pasar lista, gestionar los alumnos y otros procesos que se detallarán en el siguiente apartado sí presentan la posibilidad de mejora usando la tecnología disponible actualmente, pasando de una hoja de papel a una interface web móvil donde es más intuitivo y fácil.

1.2 Razones de elección del TFG

La razón principal de la realización de este proyecto es la experiencia personal. Soy profesor de karate de niños de entre 8 y 14 años en distintos colegios y me enfrento cada día a tareas que se pueden hacer más sencillas. Tareas como pasar lista de asistencia, realizar exámenes, gestionar los ejercicios para cada grupo de niños… Todas ellas pudiendo prescindir del papel, solo con un móvil conectado a internet.

Otra razón es que esto presenta un reto para mí. No es un proyecto estéril que será programado solo para el TFG, sino que será algo que use en mis clases a diario. También será compartida con otros compañeros para que programen sus clases y entrenamientos.

También me resulta interesante la programación web, no se ve en exceso en la carrera salvo por la asignatura optativa de *DAWE (*Desarrollo de Aplicaciones Web Enriquecidas) y creo que tiene un peso importante en las empresas de cara a la búsqueda de trabajo.

Por estas razones he decidido crear una web que me facilite a mí, a mis compañeros y alumnos ciertas tareas, que se verán con más detalle en los siguientes apartados, y que suponga un reto en cuanto a su programación.

1.3 Planteamiento del problema

Este proyecto parte de la necesidad de actualizar ciertas prácticas que son susceptibles de mejora. Dar un paso en la interacción directa entre el profesor y el alumno más allá de los canales de transferencia de información disponibles en un *Dojo*.

Se plantean dos roles, profesor y alumno.

El profesor con el uso de un dispositivo móvil, teléfono móvil o *tablet*, podrá controlar la asistencia de los alumnos con solo deslizar el dedo sobre su foto, dejando atrás el uso de hojas de papel de control de asistencia y facilitando el reconocimiento del alumno. Pudiendo sacar una foto al alumno al momento para actualizar o crear una ficha del mismo.

Se le dará la posibilidad de gestionar entrenamientos mediante un calendario para cada grupo.

Hasta ahora en el momento de realización de un examen por parte de un alumno, el profesor debía hacer una plantilla e ir rellenando los campos según trascurría el examen. Por lo cual, perdía tiempo y no podía prestar atención al examen en su totalidad. En vez de eso los campos se rellenarán con puntuaciones de una manera fácil y visual, para que no se pierda tiempo en escribir con el teclado.

Por otro lado, se dará la posibilidad al profesor de grabar una parte o la totalidad del examen para más tarde subirlo a una plataforma de almacenamiento de vídeos.

El alumno tendrá acceso a diferentes funcionalidades, algunas de ellas conectadas con el rol del profesor y otras nuevas como veremos a continuación.

Al igual que el profesor, el alumno podrá gestionar sus entrenamientos y planificarlos acorde a sus necesidades.

Tendrá a su disposición la posibilidad de controlar su progresión tanto en la consecución de cinturones como en las posiciones que obtenga en distintos campeonatos. Solo tendrá que introducir los datos y se crearán logros que podrá ver en su página principal.

Podrá grabar sus sesiones de entrenamiento y si lo desea subirlas a la plataforma de almacenamiento de vídeo para guardarlas y verlas cuantas veces quiera o compartirlas sin tener que tenerlo físicamente en el móvil. Más tarde el alumno puede notificar al profesor para que este vea la sesión.

Una de las partes importantes del karate son las *Wazas*, un encadenamiento de varios movimientos, normalmente 6 dependiendo del *Waza*. El alumno podrá sacar tantas fotos como quiera durante la composición del *Waza* y elegir las fotos que más le gusten. Una vez elegidas, la aplicación generará un carrusel de fotos. Cuando esté completo con las fotos elegidas se dispondrá de la posibilidad de subirlas a un sistema de almacenamiento en la nube. De esta manera, se tendrán guardadas las fotos más interesantes para preservarlas. Posteriormente, al igual que ocurre con las sesiones, se podrá notificar al profesor para que revise la *Waza*.

Ambos roles dispondrán un punto en común para comunicarse usando un foro en el cual puedan compartir dudas, vídeos o información interesante entre todos los usuarios.

Como la aplicación puede grabar fotos y vídeos es necesario tener todo en orden de acuerdo con la LOPD (Ley Orgánica de Protección de Datos).

La web deberá presentar una estética moderna, siendo requisito que se adapte perfectamente a diferentes dispositivos móviles.

1.4 Glosario de términos

En este apartado describiremos algunos términos que han aparecido y aparecerán en este documento para hacer más sencilla de lectura.

- TFG: Trabajo Fin de Grado.
- Dojo: Lugar donde se entrena karate.
- API: Interfaz de Programación de Aplicaciones.
- SGBD: Sistema de Gestión de Bases de Datos (por ejemplo, MySQL).
- MySQL: Programa para gestionar una BBDD.
- BBDD: Base de Datos.
- PHP: Lenguaje de programación usado en el lado del servidor.
- HTML 5: Lenguaje de programación para páginas web revisión 5.
- Hosting: Servicio de almacenamiento web.
- FTP: Protocolo de Transferencia de Archivos. Se usará para subir los archivos al servidor web.

- Dominio Web: Dirección con la que se accederá a la página web una vez este publicada en el servidor.
- Página web *Responsive*: Página web que cuenta con un formato que se adapta a la pantalla del dispositivo donde se está visualizando de manera automática.
- Swipe: Deslizamiento.

Algunos términos se explicarán mejor en los siguientes apartados.

2. Planteamiento inicial

En este apartado se detallarán cuáles son los objetivos y el alcance del proyecto, así como una estimación económica y temporal del mismo. También se llevará a cabo la gestión de riesgos, las herramientas utilizadas y la evaluación económica.

2.1 Objetivos del proyecto

Los objetivos que debe cumplir el TFG son los siguientes:

- Página web atractiva y funcional usando *Bootstrap* para ello.
- Página web *Responsive*.
- Gestión de clases y entrenamientos con ayuda de un calendario.
- Control de asistencia mediante *Swipe*.
- Posibilidad de obtener fotos del alumno usando la cámara del móvil.
- Gestión rápida del examen del alumno.
- Grabar el examen usando un *API* de almacenamiento de vídeos como *YouTube, Vimeo…*
- Zona de notificaciones.
- Control de rendimiento y logros mediante un *API* de generación de gráficas web como *Google Chart, Plotty…*
- Gestión de *Wazas* usando la cámara de fotos y un *API* de almacenamiento en la nube: *Dropbox, Google Drive…*
- Grabación por parte del alumno usando un *API* de almacenamiento de vídeos como *YouTube, Vimeo…* y compartirlo con el profesor.
- Foro mediante el uso de la *API* de *VanillaForum*.
- Creación, mantenimiento y uso de una *BBDD* usando *MySQL*.

Los objetivos personales de este TFG son los siguientes:

- Aprender a construir una página web totalmente funcional y operativa.
- Lograr una integración con distintas *API's* para hacer una página más completa.
- Aprender y mejorar en la programación en *HTML 5*.
- Tener más soltura a la hora de programar en *PHP, JavaScript* para recoger/grabar datos mediante un *SGBD*.
- Crear y mantener una BBDD desde cero.
- Aprender el procedimiento para comprar un *Dominio Web* y un *Hosting*, así como subir la web por un FTP para alojarla en el servidor comprado.

En resumen, se busca una aplicación que facilite el desempeño de ciertas tareas del profesor y del alumno durante las clases de karate. También usar todo lo aprendido en la carrera aplicado a un caso real y finalmente ganar más experiencia en el campo de la programación web de cara a la búsqueda de trabajo teniendo una web operativa que poder enseñar a modo de currículum.

2.2 Alcance del proyecto

Al principio de todo proyecto hay que definir su alcance, dicho de otra forma, clasificar las funcionalidades a implementar y desarrollar para cumplir con los objetivos establecidos previamente.

2.2.1 Fases del proyecto

- **Gestión**: es el primer paso para realizar el proyecto pues aquí se definirá el proyecto en su totalidad. Habrá que definir las tareas que lo compondrán y los requisitos necesarios para llevarlo a cabo.
- **Análisis**: aquí se definirán las funcionalidades del proyecto y las herramientas que se usarán para llevarlo a cabo.
- **Diseño**: en esta parte se diseñará la estructura de la web y las interfaces que satisfagan los puntos de gestión y análisis. Se diseñará el código que será desarrollado en las siguientes partes.
- **Implementación**: se definirán los diferentes prototipos en los cuales se va a dividir el algoritmo y su implementación. Se dividirá en los siguientes prototipos:
    1. Gestión de clases y entrenamientos.
    2. Control de asistencia mediante deslizamiento.
    3. Obtener fotos al momento y agregarlas a la ficha del alumno.
    4. Gestión rápida del examen del alumno.
    5. Grabación del examen para posterior subida al almacenamiento de vídeos elegido.
    6. Centro de notificaciones para el profesor.
    7. Control de rendimiento y logros del alumno.
    8. Gestión de *Wazas* usando la cámara, creando un carrusel y posterior subida al almacenamiento en la nube elegido.
    9. Grabación de la sesión por parte del alumno y subida al almacenamiento de vídeos elegido.
    10. Comunicación a través de un foro.
- **Plan de pruebas:** se realizará un plan de pruebas para cada prototipo implementado. Así al terminar un prototipo se testeará para verificar el correcto funcionamiento. En la fase final se realizarán pruebas finales con todos los prototipos en funcionamiento.
- **Documentación:** cierta parte de la documentación se irá trabajando a la vez que se vayan creando los prototipos y avanzando con el proyecto a fin de garantizar que todo este correctamente documentado. Al final se hará una revisión exhaustiva de la documentación para corregir y actualizar elementos que no sean correctos. Para finalizar con el proceso de documentación, se creará la presentación para la defensa del proyecto ante el tribunal.

2.2.2 Estructura de Descomposición del Trabajo (EDT)

En la siguiente imagen se detalla el diagrama EDT. Este diagrama indica cuáles son las partes en las que se divide el proyecto. Posteriormente se detallará el número de horas invertidas en cada parte.

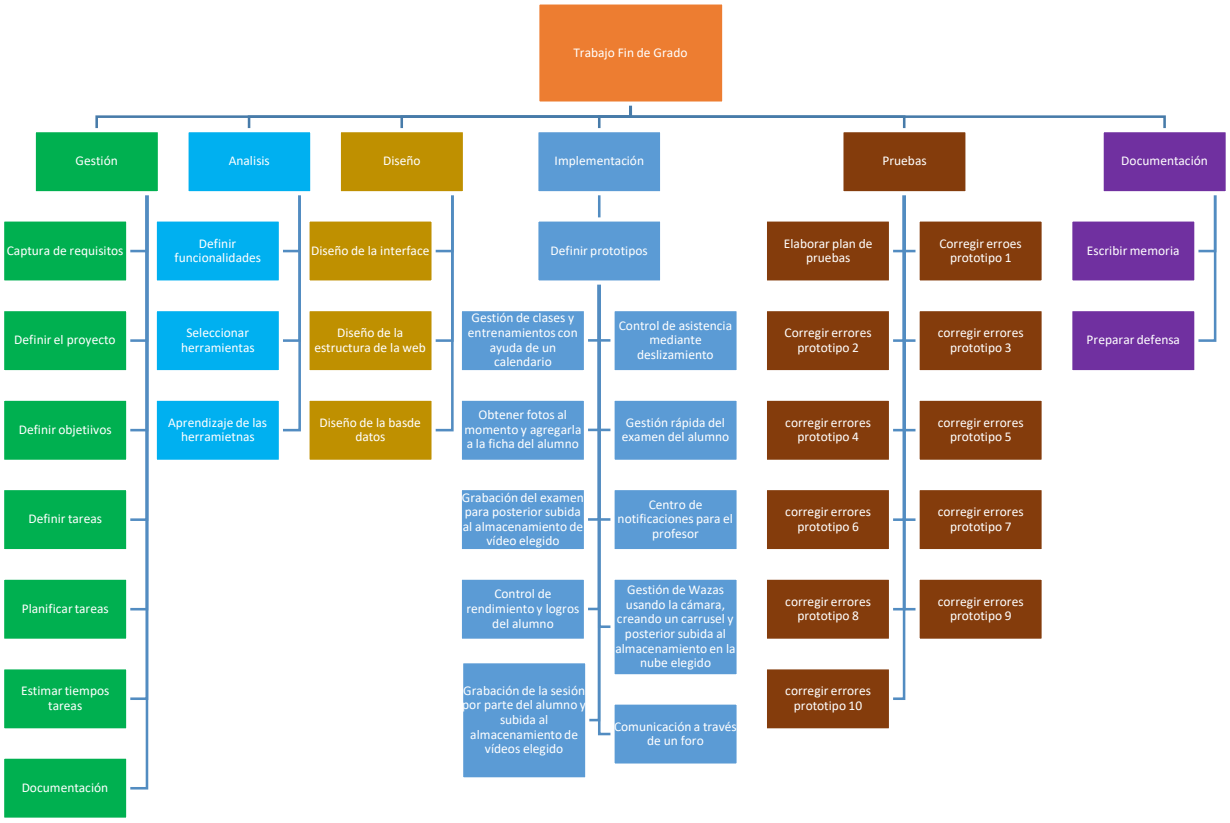


Ilustración 1: EDT

2.2.3 Tareas

En este apartado se detallarán las tareas especificadas en el *EDT*.

2.2.3.1 Gestión

2.2.3.1.1 Paquete de trabajo: [Captura de requisitos]

Duración: 3 horas

Descripción:

- Tener una idea clara de lo que va a ser el proyecto, los objetivos y las funcionalidades que debe tener.

Salidas/Entregables

- Descripción formal del trabajo.

Recursos necesarios

- Tener claro que tiene qué hacer la web.

2.2.3.1.2 Paquete de trabajo: [Definir proyecto]

Duración: 2 horas

Descripción

- Decidir cómo implementar las funcionalidades requeridas en la captura de requisitos, división de las tareas, herramientas a usar y tiempo para cada tarea entre otros.

Salidas/Entregables

- Descripción detallada del proyecto.

Recursos necesarios

- Captura de requisitos.

2.2.3.1.3: Descripción del paquete [Definir objetivos]

Duración: 1 hora

Descripción

- Definir los objetivos necesarios para la elaboración del proyecto.

Salidas/Entregables

- Objetivos del proyecto.

Recursos necesarios

Captura de requisitos y definición del proyecto.

2.2.3.1.4: Descripción del paquete [Definir tareas]

Duración: 3 Horas

Descripción

- Definir las tareas necesarias para la consecución del proyecto.

Salidas/Entregables

- Las tareas totalmente detalladas.

Recursos necesarios

- Definición de objetivos.

2.2.3.1.5: Descripción del paquete [Planificar Tareas]

Duración: 3 Horas

Descripción

- Establecer un orden de implementación de las tareas y ver si necesitan una subdivisión.

Entradas

- Tareas a planificar.

Salidas/Entregables

- La planificación de las tareas sin los tiempos.

Recursos necesarios

- Descripción de las tareas.

2.2.3.1.6: Descripción del paquete [Estimar tiempos de tareas]

Duración: 2 Horas

Descripción

- Estimar el tiempo que tardará cada tarea en ser llevada a cabo.

Entradas

- Definición de las tareas y su planificación.

Salidas/Entregables

- Las tareas con los tiempos estimados.

Recursos necesarios

- Planificación de las tareas.

2.2.3.1.7: Descripción del paquete [Documentación]

Duración: 9 Horas

Descripción

- Reunir toda la información creada hasta el momento en la parte de gestión con las tareas, su estimación de horas, objetivos y descripciones.

Salidas/Entregables

- Documento de objetivos del proyecto.

Recursos necesarios

- Todos los apartados del apartado Gestión.

2.2.3.2 *Análisis*

2.2.3.2.1: Descripción del paquete [Definir funcionalidades]

Duración: 20 Horas

Descripción

- Definir todas las funcionalidades que va a tener el proyecto.

Salidas/Entregables

- Documentación de las funcionalidades.

Recursos necesarios

- Conocimiento de las necesidades del proyecto.

2.2.3.2.2: Descripción del paquete [Seleccionar herramientas]

Duración: 3 Hora

Descripción

- Seleccionar las herramientas que mejor encajen con el desarrollo del proyecto. También encontrar tutoriales y recursos a fin de facilitar el aprendizaje de uso de las mismas.

Recursos necesarios

- Saber qué queremos de las herramientas.

2.2.3.2.3: Descripción del paquete [Aprendizaje de las herramientas]

Duración: 10 Horas

Descripción

- Practicar con las herramientas para coger soltura tanto en las herramientas de desarrollo como en las de documentación.

Salidas/Entregables

- Documentación de las funcionalidades

Recursos necesarios

- Las herramientas seleccionadas.

Precedencias:

- Seleccionar herramientas.

2.2.3.3 Diseño

2.2.3.3.1: Descripción del paquete [Diseño de la interface]

Duración: 15 Horas

Descripción

- Diseño de la parte estética de la aplicación. Creación de todas las vistas que verán los usuarios.

Salidas/Entregables

- Diseño de las interfaces.

Recursos necesarios

- Tener claro lo que se quiere mostrar en cada una.

Precedencias:

- Definición de funcionalidades.

2.2.3.3.2: Descripción del paquete [Diseño de la estructura de la web]

Duración: 30 Horas

Descripción

- Realizar un diseño de la arquitectura que va a disponer la web, así como el diseño de un algoritmo que cumpla con las funcionalidades especificadas.

Salidas/Entregables

- Algoritmo.

Recursos necesarios

- Tener claras las funcionalidades a implementar

Precedencias:

- Definición de funcionalidades.

2.2.3.3.3: Descripción del paquete [Diseño de la base de datos]

Duración: 6 Horas

Descripción

- Estudiar las tablas necesarias en la base de datos para el correcto funcionamiento de las funcionalidades de la web.

Salidas/Entregables

- Base de datos.

Recursos necesarios

- Tener claro las tablas necesarias.

Precedencias:

- Definición de funcionalidades.

2.2.3.4 Implementación

2.2.3.4.1: Descripción del paquete [Definir prototipos]

Duración: 10 Horas

Descripción

- La web se dividirá en distintos prototipos para facilitar su implementación y cada uno irá añadiendo funcionalidades al anterior.

Salidas/Entregables

- El código de cada prototipo.

Recursos necesarios

- Tener claras las funcionalidades

Precedencias:

- Análisis y diseño.

2.2.3.4.2: Descripción del paquete [Implementar Prototipo 1- Gestión de clases y entrenamientos con ayuda de un calendario]

Duración: 30 Horas

Descripción

- Tanto el profesor como el alumno podrán gestionar sus clases usando un calendario.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.3: Descripción del paquete [Implementar Prototipo 2- Control de asistencia mediante deslizamiento]

Duración: 45 Horas

Descripción

- El profesor podrá controlar la asistencia de los alumnos mediante el deslizamiento de su dedo sobre la foto del alumno.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.4: Descripción del paquete [Implementar Prototipo 3- Obtener fotos al momento y agregarla a la ficha del alumno]

Duración: 30 Horas

Descripción

- Permitir al profesor sacar una foto al momento del alumno y agregarla a la ficha del mismo.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.5: Descripción del paquete [Implementar Prototipo 4- Gestión rápida del examen del alumno]

Duración: 25 Horas

Descripción

- El profesor durante el examen puntuará al alumno a través de una interface muy sencilla e intuitiva.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.6: Descripción del paquete [Implementar Prototipo 5 – Grabación del examen para posterior subida al almacenamiento de vídeo elegido]

Duración: 20 Horas

Descripción

- El profesor podrá grabar la realización del examen de un alumno. Posteriormente podrá subir el video a una plataforma de almacenamiento de vídeos.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.7: Descripción del paquete [Implementación Prototipo 6 – Centro de notificaciones para el profesor]

Duración: 10 Horas

Descripción

- Debido a la comunicación profesor-alumno, el profesor dispondrá de un centro de notificaciones.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.8: Descripción del paquete [Implementar Prototipo 7 – Control de rendimiento y logros del alumno]

Duración: 20 Horas

Descripción

- El alumno podrá controlar sus logros y rendimientos.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.9: Descripción del paquete [Implementar Prototipo 8 – Gestión de *Wazas* usando la cámara, creando un carrusel y posterior subida al almacenamiento en la nube elegido]

Duración: 40 Horas

Descripción

- El alumno podrá fotografiarse realizando técnicas. Cuando disponga de todas las fotos, la web se las enseñará a modo de carrusel. Una vez verificadas todas las fotos el alumno las podrá subir al almacenamiento en la nube para preservarlas.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.10: Descripción del paquete [Implementación Prototipo 9 – Grabación de la sesión por parte del alumno y subida al almacenamiento de vídeos elegido]

Duración: 10 Horas

Descripción

- El alumno se podrá grabar realizando clases y/o técnicas que podrá subir al almacenamiento de vídeos elegido para posteriormente notificárselo al profesor si así desea.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.4.11: Descripción del paquete [Implementación Prototipo 10 – Comunicación a través de un foro]

Duración: 20 Horas

Descripción

- Tanto el profesor como el alumno dispondrán de un lugar de encuentro donde compartir conocimiento u ocio. Para ello se implementará un foro.

Salidas/Entregables

- El código implementado.

Recursos necesarios

- Prototipo definido.

Precedencias:

- Análisis y diseño.

2.2.3.5 Plan de pruebas

2.2.3.5.1: Descripción del paquete [Elaborar un plan de pruebas por cada prototipo]

Duración: 15 Horas

Descripción

- Desarrollar un plan de pruebas para cada prototipo que se ajuste a lo implementado para poder encontrar errores y corregirlos.

Salidas/Entregables

- Será necesario el código del prototipo

Salidas/Entregables

- Documentación de las pruebas

Recursos necesarios

- El código implementado y conocer las pruebas necesarias para cada prototipo.

Precedencias:

- Implementación.

2.2.3.5.2: Descripción del paquete [Corregir errores Prototipo 1]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 1

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.3: Descripción del paquete [Corregir errores Prototipo 2]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 2

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.4: Descripción del paquete [Corregir errores Prototipo 3]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 3

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.5: Descripción del paquete [Corregir errores Prototipo 4]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 4

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.6: Descripción del paquete [Corregir errores Prototipo 5]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 5

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.7: Descripción del paquete [Corregir errores Prototipo 6]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 6

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.8: Descripción del paquete [Corregir errores Prototipo 7]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 7

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.9: Descripción del paquete [Corregir errores Prototipo 8]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 8

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.10: Descripción del paquete [Corregir errores Prototipo 9]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 9

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.5.4: Descripción del paquete [Corregir errores Prototipo 10]

Duración: 5 Horas

Descripción

- Corregir errores identificados en el prototipo 10

Salidas/Entregables

- Documentación de los errores encontrados y resueltos.

Recursos necesarios

- Código implementado.

Precedencias:

- Implementación y plan de pruebas.

2.2.3.6 Documentación

2.2.3.6.1: Descripción del paquete [Escribir memoria]

Duración: 35 Horas

Descripción

- Escribir la memoria del proyecto. Está será obtenida recabando toda la documentación junto con las tareas, diseños y plan de pruebas.

Salidas/Entregables

- La memoria.

Recursos necesarios

- Proyecto y toda la documentación recabada a lo largo de todo el proyecto.

Precedencias:

- Todos los puntos anteriores.

2.2.3.6.2: Descripción del paquete [Preparar la defensa]

Duración: 35 Horas

Descripción

- Preparar la defensa del proyecto

Salidas/Entregables

- La defensa

Recursos necesarios

- La documentación y el proyecto.

Precedencias:

- Todos los puntos anteriores.

2.3 Planificación temporal

En este apartado se recopilarán todas las fases y tareas anteriores y se mostrarán en una tabla. Se calculará el tiempo necesario para con consecución del proyecto y se mostrarán distintos gráficos sobre la planificación temporal.

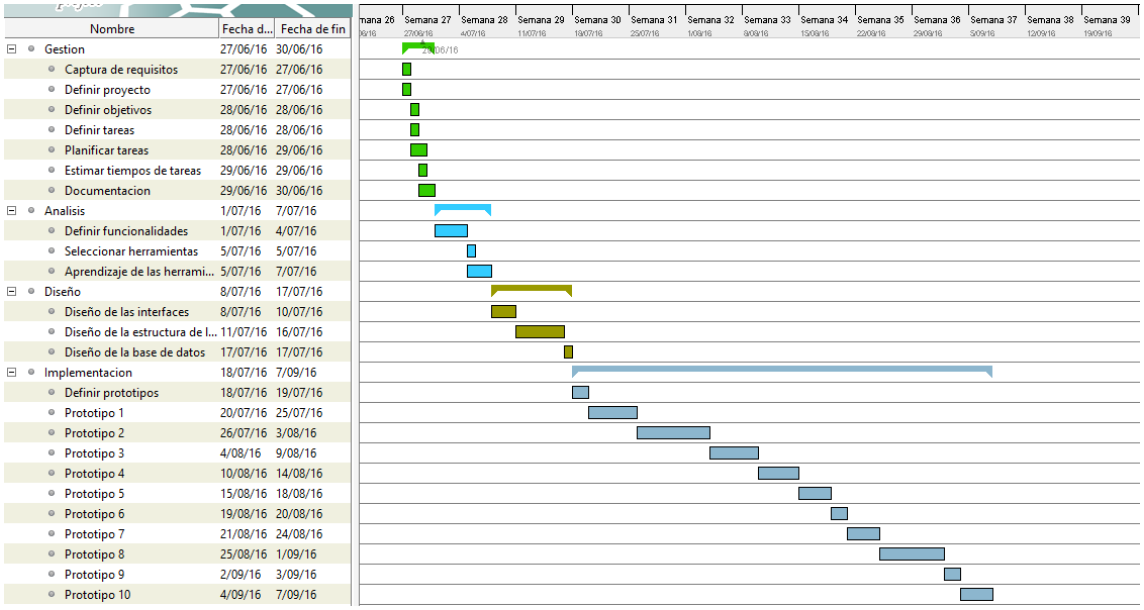
Tareas	Estimación de horas
1-Gestión	**23**
1.1 Captura de requisitos	3
1.2 Definir proyecto	2
1.3 Definir objetivos	1
1.4 Definir tareas	3
1.5 Planificar tareas	3
1.6 Estimar tiempos de tareas	2
1.7 Documentación	9
2-Análisis	**33**
2.1 Definir funcionalidades	20
2.2 Seleccionar herramientas	3
2.3 Aprendizaje de las herramientas	10
3- Diseño	**51**
3.1 Diseño de las interfaces	15
3.2 Diseño de la estructura de la web	30
3.3 Diseño de la base de datos	6
4-Implementación	**260**
4.1 Definir prototipos	10

4.2 Implementar prototipo 1	30
4.3 Implementar prototipo 2	45
4.4 Implementar prototipo 3	30
4.5 Implementar prototipo 4	25
4.6 Implementar prototipo 5	20
4.7 Implementar prototipo 6	10
4.8 Implementar prototipo 7	20
4.9 Implementar prototipo 8	40
4.10 Implementar prototipo 9	10
4.11 Implementar prototipo 10	20
5-Plan de pruebas	**65**
5.1 Elaborar plan de pruebas	15
5.2 Corregir errores prototipo 1	5
5.3 Corregir errores prototipo 2	5
5.4 Corregir errores prototipo 3	5
5.5 Corregir errores prototipo 4	5
5.6 Corregir errores prototipo 5	5
5.7 Corregir errores prototipo 6	5
5.8 Corregir errores prototipo 7	5
5.9 Corregir errores prototipo 8	5
5.10 Corregir errores prototipo 9	5
5.11 Corregir errores prototipo 10	5
6-Documentación	**70**
6.1 Escribir memoria	35
6.2 Preparar la defensa	35
TOTAL	**502**

Tabla 1 Estimación temporal

Se estima que se inviertan en el trabajo un total de 35 horas semanales como norma, pero serán variables a fin de tener el trabajo hecho para la segunda semana de septiembre. Por tanto, el trabajo debería estar completo en aprox. 12 semanas. Al ser un trabajo personal está sujeto a mi disponibilidad horaria, pudiendo hacer más horas semanales si es necesario. El reparto de horas es semanal puesto que hacerlo diario es más complicado debido a la flexibilidad que presenta el trabajar por cuenta propia. Se trabajarán jornadas más largas a veces y se trabajará fines de semana, festivos y laborables indistintamente.

Para mostrar algo de claridad se ofrece a continuación un diagrama de *GANTT* adaptándolo a las 35 horas semanales de trabajo.



Nombre	Fecha d...	Fecha de fin
Gestion	27/06/16	30/06/16
Captura de requisitos	27/06/16	27/06/16
Definir proyecto	27/06/16	27/06/16
Definir objetivos	28/06/16	28/06/16
Definir tareas	28/06/16	28/06/16
Planificar tareas	28/06/16	29/06/16
Estimar tiempos de tareas	29/06/16	29/06/16
Documentacion	29/06/16	30/06/16
Analisis	1/07/16	7/07/16
Definir funcionalidades	1/07/16	4/07/16
Seleccionar herramientas	5/07/16	5/07/16
Aprendizaje de las herrami...	5/07/16	7/07/16
Diseño	8/07/16	17/07/16
Diseño de las interfaces	8/07/16	10/07/16
Diseño de la estructura de l...	11/07/16	16/07/16
Diseño de la base de datos	17/07/16	17/07/16
Implementacion	18/07/16	7/09/16
Definir prototipos	18/07/16	19/07/16
Prototipo 1	20/07/16	25/07/16
Prototipo 2	26/07/16	3/08/16
Prototipo 3	4/08/16	9/08/16
Prototipo 4	10/08/16	14/08/16
Prototipo 5	15/08/16	18/08/16
Prototipo 6	19/08/16	20/08/16
Prototipo 7	21/08/16	24/08/16
Prototipo 8	25/08/16	1/09/16
Prototipo 9	2/09/16	3/09/16
Prototipo 10	4/09/16	7/09/16

Ilustración 2 Gantt 1

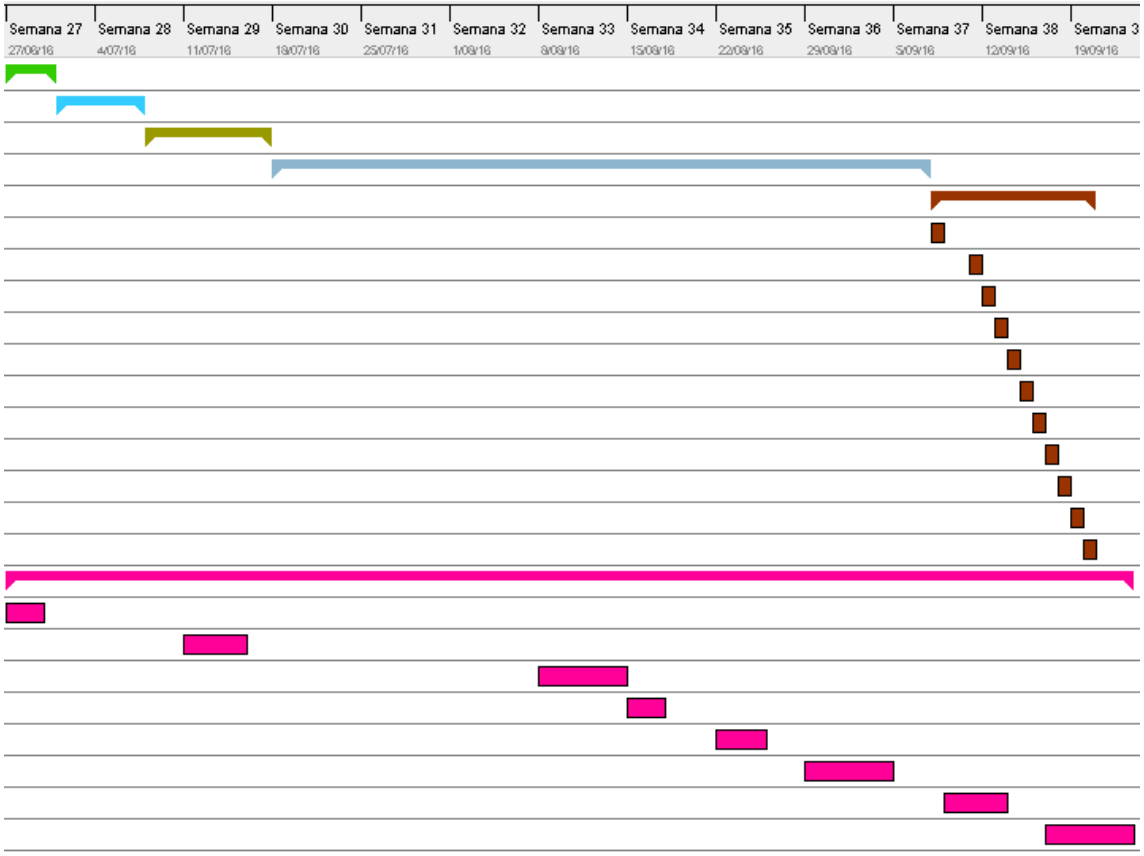


Ilustración 3 Gantt II

*NOTA: la sección rosa hace referencia a la documentación.

2.4 Arquitectura

Para tener una mejor visión del proyecto se va a dividir esta sección en tres apartados. En el primero se verá la arquitectura interna de la web, en el segundo la arquitectura externa de la web y por último la arquitectura cliente/servidor.

2.4.1 Arquitectura interna

En la siguiente figura aparece reflejada la arquitectura interna de la web, la cual se detallará más a continuación:

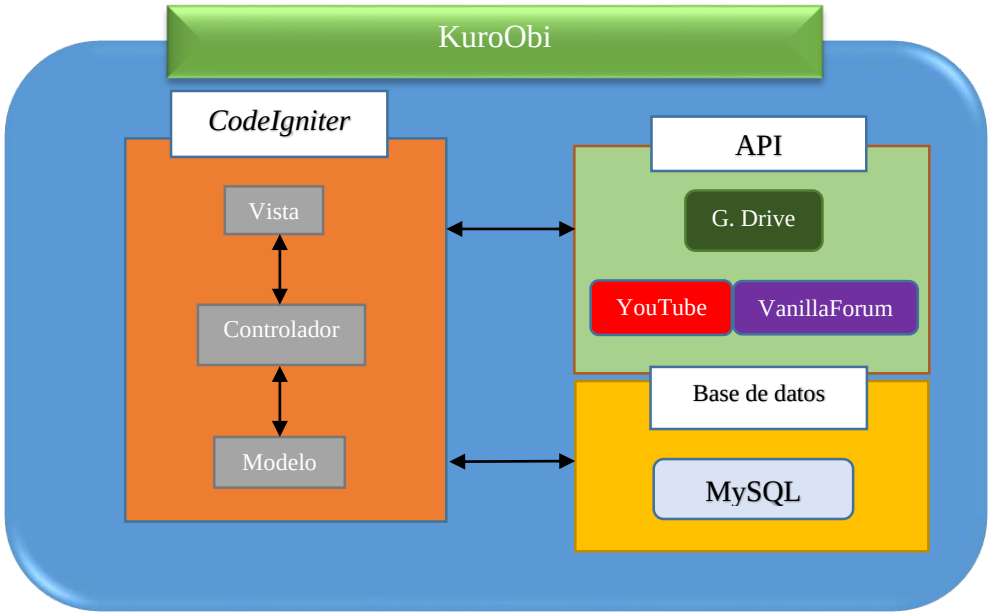


Ilustración 4 Arquitectura interna de la web

***CodeIgniter*:** es el *framework* encargado de separar la web en Modelo-Vista-Controlador (MVC o Model-View-Controller) que se explica con detalle en apartados posteriores.

Vista: La interface de la web donde el usuario introduce y recibe datos.

Controlador: Se encarga de gestionar los datos que introduce el usuario.

Modelo: Se encarga de comunicarse con la base de datos.

MySQL: Es el sistema encargado de gestionar la base de datos.

Base de datos: Lugar donde se almacenan los datos del usuario.

API: Permite acceder sistemas y funcionalidades creadas por terceros.

YouTube: Permite subir vídeos a la plataforma de vídeo digital.

G. Drive (Google Drive): Permite subir fotos al almacenamiento en la nube de google.

VanillaForum: Permite insertar un foro para permitir la comunicación entre usuarios a través de la web.

2.4.2 Arquitectura externa

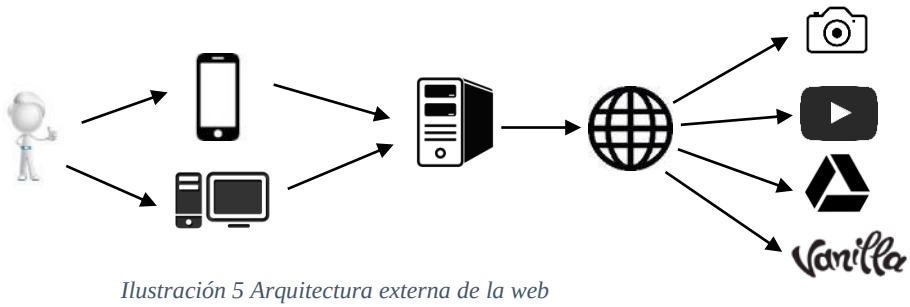


Ilustración 5 Arquitectura externa de la web

Como se observa en la ilustración anterior, el usuario accede a la web con un móvil o con un ordenador a través del navegador web que carga los datos en el servidor. Una vez se muestra la web, el usuario puede trabajar con ella y por ejemplo: hacer uso de la cámara de fotos para grabar exámenes, subir vídeos a *YouTube*, guardar sus fotos en *Google Drive* o hablar con otros usuarios a través del foro proporcionado por *Vanilla*.

2.4.3 Arquitectura cliente-servidor

La web y la base de datos se alojarán en un servidor y será accesible por el usuario mediante el uso de ordenador, móvil o *tablet* (cliente).

Una descripción más formal: *"La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta." [1]*

En la siguiente imagen se puede apreciar esta arquitectura de una manera sencilla.

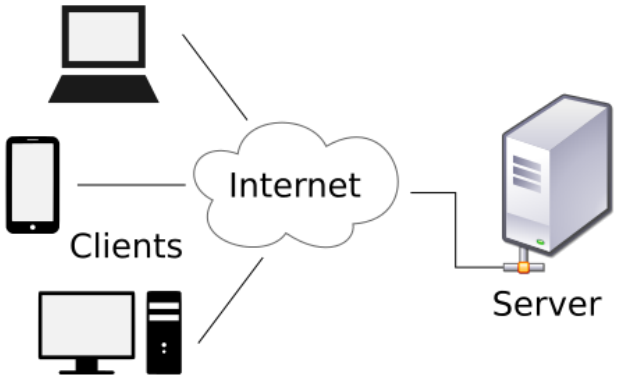


Ilustración 6 Cliente/servidor[2]

Por otro lado, al ser una página web presenta 3 capas o niveles. Un nivel representa un elemento que procesa y trata información. Toda la parte visual de la web se encuentra en el nivel 1, mientras que la lógica se procesa en el nivel 2 y la base de datos en el nivel 3.

2.5 Herramientas

Para la creación de la página web, así como su lógica, se usará las siguientes herramientas:

2.5.1 Hardware:

- Portátil
- Ordenador de sobremesa
- Móvil
- Tablet

2.5.2 Software:

- **Sublime Text 3**: un editor de textos que proporciona ayuda visual a la hora de programar tanto en *PHP* como en *HTML*.

Ilustración 7 Sublime Text 3

- **Google Chrome**: navegador de internet. Necesario para controlar los avances y acceder a la web.

Ilustración 8 Google Chrome

- **Wamp Server**: esta aplicación permite emular un servidor apache en el propio ordenador, es necesario para poder interpretar el *PHP*. También posee *MySQL;* un gestor de base de datos donde poder almacenar toda la información necesaria para la página web.

Ilustración 9 Wamp Server

- **Microsoft Office**: herramienta ofimática para la documentación y memoria.

Ilustración 10 Microsoft Office

- **SourceTree**: este programa permite controlar los cambios en los ficheros para luego subirlos al servidor de *BitBucket*. De esta manera, los archivos siempre están actualizados y se tiene un gestor de cambios.

Ilustración 11 SourceTree

- **Visual Paradigm**: programa usado para generar los diagramas de secuencia, modelo de dominio y casos de uso.

Ilustración 12 Visual Paradigm

2.5.3 Lenguajes de programación

- **PHP**: *"PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico" [4]*. Este código se inserta dentro el HTML escogido ya que es interpretado por la mayoría de los servidores web.

Ilustración 13 PHP

- **HTML**: *"HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web." [5]*. Se llama lenguaje de marcas dado que todo el código se basa en etiquetas. Este lenguaje está reconocido y aceptado por todas las empresas relacionadas con internet, por eso lo he elegido.
- **JavaScript**: se usa para darle más funcionalidad al código *HTML*. Todos los navegadores interpretan JavaScript.

2.5.4 Frameworks

- **Bootstrap**: es un framework de *CSS*. Añade multitud de mejoras visuales a la estética de la web haciéndola más atractiva y moderna.

Ilustración 14 Bootstrap

- ***CodeIgniter***: esta herramienta nos facilita la separación MVC (Modelo Vista Controlador). El patrón arquitectural MVC ayuda a estructurar el código de la aplicación para desacoplar aspectos de control, presentación al usuario y persistencia de los datos, con el fin de facilitar el desarrollo y mantenimiento de la aplicación en el tiempo. Las clases que implementan el apartado *View* (Vista)se encargan de implementar las funcionalidades y aspecto visual relacionados con el interfaz de usuario. Las clases que implementan el apartado *Controller* (Controlador) son las encargadas de la lógica y de procesar los datos. Las clases que implementan el apartado *Model* (Modelo) son las encargadas de mantener la persistencia de los datos y ofrecerlos cuando el controlador lo requiera.

Ilustración 15 CodeIgniter

2.6 Gestión de riesgos

Todo proyecto de cierta índole tiene unos riesgos asociados que conviene identificar para poder evitarlos (en la medida de lo posible) o, en caso de ser imposibles de evitar, estar preparado para actuar y solucionarlos lo más rápido posible y con la mayor eficacia.

En esta sección se detallarán los posibles riesgos detectados, la probabilidad de que ocurran, el impacto que supondría en el proyecto, así como las consecuencias que causarían. Se detallarán también las medidas de prevención tomadas, así como el plan de contingencia que se aplicará para arreglar el problema.

2.6.1 Planificación Incorrecta del tiempo del proyecto.

- ***Descripción:*** Al ser una de las primeras planificaciones que se realizan para un proyecto tan grande, es muy probable que las fechas y tiempos marcados en la planificación no sean correctos. Siendo estos tiempos demasiado optimistas, produciendo que la fecha de entrega no sea correcta.
- ***Probabilidad****:* Alta.
- ***Impacto:*** Alto.
- ***Consecuencia***: Una mala planificación del proyecto retrasaría todos los procesos y con ello la fecha de entrega se vería afectada.
- ***Prevención:*** La manera de prevenir este problema será hacer plazos de entrega más amplios para así tener margen de maniobra.
- ***Plan de contingencia:*** Se aumentarán las horas diarias invertidas a fin de encauzar los flujos temporales de nuevo para que la fecha final se vea afectada lo menos posible.

2.6.2 Baja por accidente o enfermedad

- ***Descripción:*** Debido al clima cambiante que sufrimos ahora en verano es posible caer enfermo o también sufrir un accidente debido al periodo ocioso que representa esta estación.
- ***Probabilidad****:* Baja.
- ***Impacto:*** Media.
- ***Consecuencia***: Sin ponernos en casos graves y teniendo en cuenta las enfermedades y accidentes más comunes, las consecuencias serían de pérdida de algunas horas de trabajo. En caso de problemas más graves el retraso o pérdida de horas será mucho mayor.
- ***Prevención:*** No existe una prevención contra este tipo de problemas pues son aleatorios. Si bien se pueden tener en cuenta a la hora de planificar tareas dándoles un poco más de margen.
- ***Plan de contingencia:*** Se aumentarán las horas de trabajo de otros días a fin de compensar las horas perdidas.

2.6.3 Pérdidas totales o parciales de documentos/ficheros

- ***Descripción:*** Es posible que los ficheros y/o documentos sufran algún tipo de corrupción o perdida que imposibilite su lectura y trabajar con ellos.
- ***Probabilidad****:* Media
- ***Impacto:*** Baja.
- ***Consecuencia***: Perdida parcial del trabajo.

- ***Prevención:*** Tener diferentes copias de seguridad alojadas en diferentes almacenamientos tanto en la nube (Dropbox, OneDrive, Google Drive…) como físicos (Disco duro, Disco duro externo.).
- ***Plan de contingencia:*** En caso de que produjera este accidente, se procedería a restaurar una copia de seguridad en lugar del afectado para seguir trabajando.

2.6.4 Sufrir fallos en el ordenador de trabajo.

- ***Descripción:*** Es posible que el entorno de trabajo sufra daños y deje de funcionar.
- ***Probabilidad****:* Baja.
- ***Impacto:*** Bajo
- ***Consecuencia***: Perdida del entorno de trabajo.
- ***Prevención:*** Tener más dispositivos para poder seguir trabajando en caso de que un equipo falle.
- ***Plan de contingencia:*** Usar otro equipo para seguir con el trabajo hasta que el principal esté en condiciones de volver a ser usado.

2.6.5 Problemas a la hora de implementar algoritmo

- ***Descripción:*** Es muy posible que aparezcan problemas a la hora de implementar un algoritmo nuevo.
- ***Probabilidad****:* Alta.
- ***Impacto:*** Medio
- ***Consecuencia***: Retraso en las siguientes tareas.
- ***Prevención***: Dedicar el tiempo necesario a diseñar un código que sea fácil de implementar. También dedicar tiempo a la búsqueda de recursos para ayudarme con la tarea.
- ***Plan de contingencia:*** Dedicar más horas a la implementación a fin de no retrasar las demás tareas.

2.6.6 Problemas a la hora de implementar prototipos debido a cambios en las *API.*

- ***Descripción:*** Es posible que, durante la implementación de alguna funcionalidad o una vez ya implementada, la *API* cambie dejando inservible el código actual.
- ***Probabilidad****:* Baja
- ***Impacto:*** Medio

- ***Consecuencia***: Retraso en las demás tareas pudiendo afectar a la fecha de entrega del proyecto.
- ***Prevención***: Estar al día de todos los posibles cambios y actualizaciones que sufra la *API*.
- ***Plan de contingencia:*** Considerar si los cambios de la *API* son demasiado costosos de asumir a la hora de actualizar el código. Si el cambio obliga a demasiados cambios se puede buscar un servicio similar que no requiera tanto coste horario. Además, aumentar las horas de trabajo a fin de no retrasar las demás tareas y la fecha de entrega.

2.6.7 Problemas personales.

- ***Descripción:*** Aparición de problemas de índole personal que no permitan dedicar las horas diarias al proyecto.
- ***Probabilidad****:* Baja.
- ***Impacto:*** Medio.
- ***Consecuencia***: Retraso de las fechas establecidas para las tareas.
- ***Prevención***: No hay una determinada pues depende del tipo de problema que se tenga.
- ***Plan de contingencia:*** Aumentar las horas de trabajo para compensar las posibles horas perdidas.

2.7 Evaluación económica

En este punto se verá la evaluación económica del proyecto, su coste total y las diferentes formas de recuperar la inversión realizada.

Se dividirá en los siguientes apartados:

- **Personal:**
  Para la elaboración del trabajo se necesita el trabajo de un desarrollador. Teniendo en cuenta que un programador cobra 25€/hora y que el proyecto dura 502 horas, el coste en cuanto al personal asciende a 12.550€.
- **Hardware:**
  - **Ordenador de mesa:** valorado en 1000€ con una vida media de 5 años. Haciendo uso prácticamente total del ordenador durante la elaboración del TFG, tanto para desarrollo como para documentación. Con un uso dedicado al TFG del 33% a lo largo del proyecto.
    - Amortización anual: 1000/5=200€.
    - Gasto del ordenador: ((200* 14semanas) *0.33/52semanas=**17.76 euros.**
  - ***tablet* Asus Nexus 7**: Valorada en 600€ con una vida media de 3 años y un uso del 20% a lo largo del proyecto.

- Amortización anual= 600/3=200€.
- Gastos de la *tablet*: ((200*14 semanas) *0.2/52 semanas = **10.7 euros.**

- **Gastos totales de hardware: 28,46€**

- **Software:**
  - Licencia de Windows 10 gratuita mediante *DreamSpark* (portal de la universidad junto a Microsoft que permite a los estudiantes descarga de software).
  - Wamp: licencia gratuita.
  - Gantt Project: licencia gratuita.
  - Visual Paradigm: licencia gratuita.
  - Sublime Text: licencia gratuita
  - Google Chrome: licencia gratuita.

  - **Gastos totales de software: 0€**

- **Hosting**:
  Compra del servidor para almacenar y el dominio "kuroobi.es".
  El servicio se compra por un periodo de un año.

  - **Gastos totales de hosting: 90€**

Gastos	
Hardware	28,46
Ordenador	17,76
tablet	10,7
Software	0
Personal	12.550
Hosting	90
TOTAL	**12.696,92**

Tabla 2 Tabla de gastos

2.7.1 Recuperar la inversión

Para recuperar la inversión se poseen diferentes opciones si bien no se tiene clara cuál escoger debido a la inexperiencia en este campo. Vender una web se antoja más complicado que una aplicación que puedes colocar en un *market*. En las webs se pueden explorar otras opciones que veremos a continuación

- Financiación mediante anuncios: Una de las opciones que se barajan para recuperar la inversión es la inclusión de anuncios no invasivos (que no sean pop-ups ni contengan sonido) siempre con temática deportiva en los márgenes de la web. Los anuncios generan dinero según el número de pulsaciones de ratón que se efectúen sobre ellos y también dependen del tráfico de la página. Esta opción

es interesante pero no se espera que el tráfico de usuarios será muy alto pues un profesor gestiona muchos alumnos y es para un público muy restringido.

- Financiación por cliente: Otra opción sería vender la aplicación al cliente directamente por un precio determinado a modo de licencia bien anual bien permanente. Un precio aproximado por cliente sería de 200€. Esto le daría acceso a la aplicación con un uso ilimitado y también le daría acceso asistencia directa en caso de fallos o errores.
  Vendiéndola a 200€ por persona seria necesario venderles la web a unas 63 personas para empezar a rentabilizar la inversión.

- Financiación por empresa: Este caso sería similar al anterior, pero en vez de ir al cliente ir a la empresa o colegio que le contrata. Siendo profesor de karate es muy probable que este contratado por el colegio o una empresa de deportes extraescolares. La web se puede adaptar a distintos deportes con unos pequeños ajustes. Es muy posible que, si se vende a una empresa con más deportes aparte del karate, hubiera que diseñar algunas funciones nuevas que se ajusten más que las que ya están diseñadas. Por ejemplo, en futbol la funcionalidad de realizar exámenes no tiene mucho sentido y cobraría más interés una donde controlar diferentes estadísticas del jugador: disparo, carrera, goles…
  Esto haría que el precio de la aplicación se incrementase, pero también los posibles clientes, lo que aumentaría la expansión de la web.
  El precio para una empresa dependería de cuantos trabajadores van a usarla y si es necesario ampliar la web a otros deportes y otras funcionalidades.

3. Antecedentes

En este capítulo se pretende dotar al proyecto de un trasfondo, donde se vea perfectamente para qué y por qué se quiere realizar este proyecto.

3.1 Historia

Soy profesor de karate desde hace unos 10 años al momento de la redacción de este documento lo que nos sitúa en 2006. Desde el primer día he tenido que gestionar los apartados no estrictamente del karate de manera poco eficiente a mi modo de ver. Estos apartados incluyen: gestionar alumnos, pasar lista, evaluar exámenes… y otros comentados en los puntos anteriores del documento.

Por ese motivo unido a los conocimientos informáticos que poseo y el despliegue tecnológico actual se puede afrontar una renovación tecnológica para estos procesos.

3.2 Situación actual

Como hemos visto en apartados anteriores se requiere mejorar ciertos procesos para hacerlos más fáciles y rápidos.

Para ello se hará uso del dispositivo móvil ya sea móvil o *tablet*, permitiendo agilizar procesos in situ.

3.3 Estudio de diferentes alternativas.

Se estudiaron muchas aplicaciones para gestionar entrenamientos para dispositivos móviles.

En primer lugar, se buscó karate en la “Google Play Store” para ver las aplicaciones listadas en la tienda de Google. Quitando los juegos o aplicaciones que no tienen que ver estrictamente con el karate, el número de aplicaciones se redujo a 8.

Estas aplicaciones funcionan todas de manera muy similar siendo todas para aprender karate. Constan de un menú donde elegir una técnica o un Kata, y mediante fotos o videos enseñan su ejecución. De esta manera, este tipo de aplicaciones no representan una alternativa a este proyecto.

A continuación, se buscaron las aplicaciones que gestionan entrenamientos de una manera más general. Este tipo de aplicaciones se basan casi de manera exclusiva en gestión de entrenamientos para gimnasios, dando rutinas de pesas o ejercicios para la mejora cardiovascular.

También se buscaron los términos “control de alumnos” y “control de clases” con el fin de encontrar alternativas al TFG. En el primer término encontramos aplicaciones

educacionales para controlar alumnos. En este apartado si hay aplicaciones donde se puede pasar asistencia a los alumnos y controlar ciertos varemos (notas, puntualidad…). Respecto al segundo término, dio cientos de resultados, pero de temáticas dispares y ninguno centrado al control de clases en el sentido estricto.

Después de revisar todas estas aplicaciones se constata que no hay ninguna que aúne todos los requisitos funcionales especificados para este TFG.

Se han mencionado un sector de aplicaciones que, sí gestionan ciertos valores para el control de alumnos, pero seguiría faltando que estuvieran enfocadas al karate. Y las que están enfocadas no controlan a los usuarios.

Este proyecto no se basa tanto en el aprendizaje del karate como se muestran en las apps sino en que un profesor o alumno puedan gestionar clases o así mismo respectivamente.

Por esta razón no se ha visto una alternativa que enfoque el karate desde la gestión de sus clases y seguimiento de alumnos.

4. Captura de requisitos

En este apartado veremos los requisitos previos de nuevo para conectar las peticiones del cliente con los diagramas de casos de uso y posteriormente con el diagrama de modelo de dominio.

4.1 Requisitos previos

Para los siguientes diagramas es importante conocer los requisitos que debe cumplir la aplicación. Ya los hemos visto en anteriores puntos, pero los mencionaremos una vez más para refrescarlos y poder ver los siguientes apartados más claramente.

La aplicación tendrá dos roles, profesor y alumno. El profesor podrá dar de alta tantos alumnos como desee que se irán agrupando por colegio, curso y grupo. El profesor podrá gestionar alumnos de manera tradicional, dar de alta, baja y modificaros. A la hora de pasar lista el profesor lo realizará deslizando el dedo a izquierda o derecha sobre la foto del alumno.

El profesor podrá crear un calendario de entrenamientos para los diferentes grupos de alumnos que gestione.

Por otro lado, el profesor podrá realizar exámenes puntuándolos de manera rápida y muy visual sin perder ni un ápice del examen del alumno. También podrá grabar el examen para verlo más adelante, para ello lo podrá subir a una página de almacenamiento de vídeos online.

Por el lado del alumno tenemos otras funcionalidades, en primer lugar, una que comparte con el profesor será la posibilidad de crear calendarios de entrenamientos, aunque esta vez serán personales y no para un grupo de alumnos.

El alumno podrá controlar su progreso en campeonatos y en clase, mediante la consecución de cinturones.

Otra funcionalidad disponible para el alumno sería la posibilidad de grabarse en vídeo realizando un ejercicio o entrenamiento para después subirlo a la página de almacenamiento. Una vez subido podrá notificar al profesor para que lo visualice cuando crea oportuno.

También dispondrá de una herramienta para crear sus *Wazas*, creando un carrusel de fotos que podrá editar como crea conveniente hasta tener las fotos que más le gusten. Una vez que tenga las fotos, las podrá subir a una web de almacenamiento online y notificar al profesor como ocurre con los vídeos.

Dado que el alumno puede subir vídeos y fotos y mandarle notificaciones al profesor, este último contará con un área de notificaciones donde ver los últimos acontecimientos de sus alumnos.

Por último, se quiere que tanto alumnos como profesores tengan un punto de reunión donde poder tratar temas de diversos tipos ya sean sobre karate o no. Para ello se les proporcionará un foro donde poder conversar tranquilamente.

4.2 Casos de uso

Después de realizar la captura de requisitos es necesario detallar las funciones que guardan relación directa con los requisitos.

A continuación, se van a mostrar los casos de uso extraídos de las especificaciones de usuario. Se describirá la jerarquía de actores implicados con una pequeña descripción. Después se mostrará el modelo de dominio.

Los casos de uso extendidos se mostrarán en el Anexo I. En ellos se detallarán con más precisión los casos de uso.

4.2.1 Jerarquía de actores

La aplicación tiene 3 actores:

Usuario no-identificado: Este actor es el usuario que no se ha registrado en la web. Tendrá acceso al área pública, un área donde ver la historia del karate, vídeos y fotos. Llegado el momento se podrá registrar como alumno o profesor.

Usuario Profesor: Este actor puede generar calendarios de entrenamiento, puede dar de alta/modificar/borrar alumnos. El profesor podrá realizar exámenes a los alumnos y también podrá grabar en vídeo el examen y subirlo a internet. Tendrá una herramienta para pasar lista de manera rápida y visual.

Usuario Alumno: El alumno tiene acceso a la web y a un apartado específico en ella donde podrá controlar sus entrenamientos de manera autónoma, siempre en contacto con el profesor, pero pudiendo auto gestionarse. Podrá también controlar su progreso en campeonatos o exámenes. Tiene la posibilidad de grabar o fotografiar sesiones que podrá enviar al profesor para que este controle su evolución.

Tanto el profesor como el alumno podrán usar un foro para comunicarse.

A continuación, se muestra la imagen de la jerarquía de actores.

Ilustración 16 Jerarquía de actores

4.2.2 Modelo de casos de uso

A continuación, se ilustra el modelo de casos de uso.

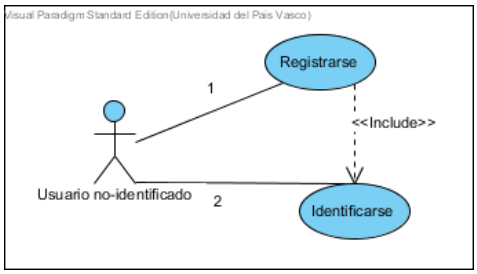


Ilustración 17 C.U. Usuario no-identificado

4.2.2.1 Casos del uso del profesor.

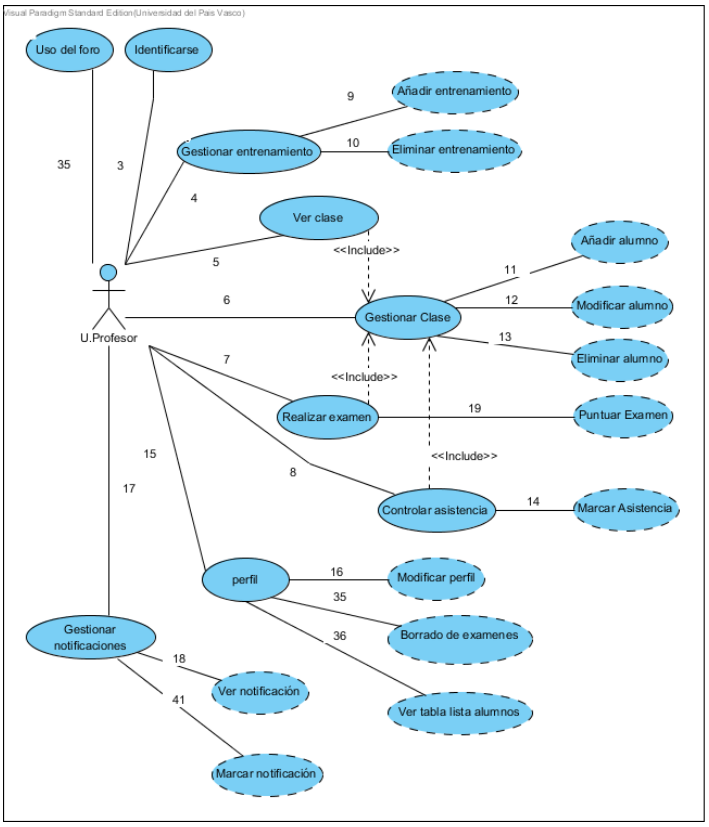


Ilustración 18 C.U. U. Profesor

4.2.2.2 Casos de uso del alumno

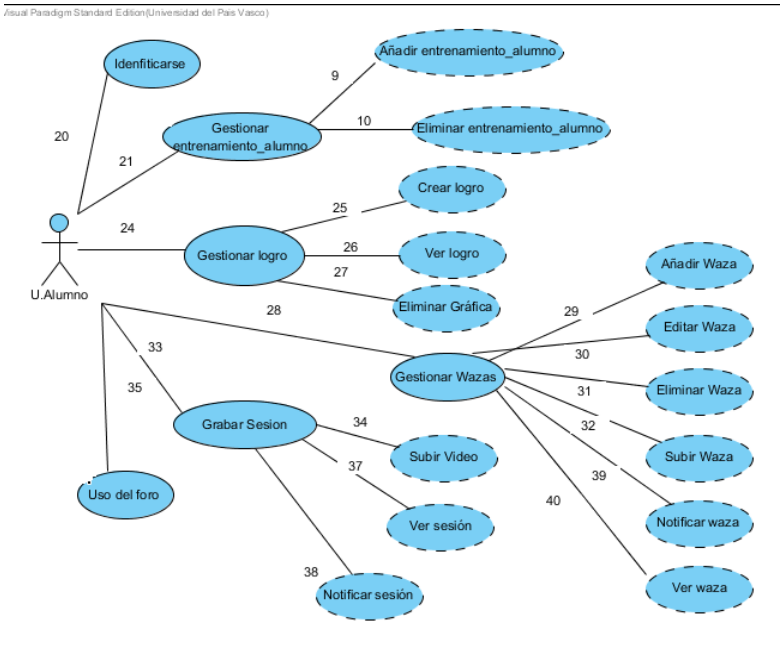


Ilustración 19 C.U. U. Alumno

1 Registro

Permite al usuario darse de alta en la web y elegir el rol. Una vez registrado tendrá acceso a nuevas funcionalidades.

2-3-20 Identificarse

Permite al usuario previamente registrado y con un rol, profesor o alumno, entrar en la web y poder acceder a las funcionalidades diseñadas para cada uno de los papeles.

4-21 Gestionar entrenamiento/Gestionar entrenamiento_alumno

Permite al profesor y al alumno crear un plan de entrenamiento. En el caso del profesor podrá hacerlo para las clases que tenga y en el del alumno para uso propio.

9-22 Añadir entrenamiento/Añadir entrenamiento_alumno.

Permite a ambos roles colocar una ficha con el entrenamiento detallado.

10-23 Eliminar entrenamiento/Eliminar entrenamiento_alumno

Permite eliminar las etiquetas de entrenamiento.

5 Ver clase

Permite al profesor ver las fotos de los alumnos de una clase.

6 Gestionar clase

El profesor entrará en esta área para gestionar todo lo relativo a una clase. Podrá verla, controlar la asistencia de todos los alumnos de la clase o realizarles un examen.

11 -12-13 Añadir/modificar/eliminar alumno

Permite al profesor añadir, modificar y/o eliminar a un alumno.

7 Realizar examen

Da la posibilidad al profesor de realizar un examen a todos los alumnos de una clase.

19 Puntuar examen

Una vez hecho el examen, el profesor deberá puntuarlo.

8 Controlar asistencia

Funcionalidad que permite al profesor controlar la asistencia de los alumnos de una clase mediante el deslizamiento sobre la foto del alumno.

14 Marcar asistencia

Dependiendo de la orientación del deslizamiento, marcará al alumno como presente o no presente.

15 Perfil

El profesor podrá controlar todo lo relacionado con sus datos y modificarlos (nº 16)

17-18 Gestionar Notificaciones/verlas

Si el alumno le manda un vídeo o fotos al profesor, éste recibirá una notificación de manera visual en su panel donde podrá ver de qué se trata.

35 Uso del foro

Da la posibilidad al profesor y al alumno de comunicarse entre ellos a través de un foro.

24-25-26-27 Logros y progresiones (Añadir/eliminar/ver logros)

El alumno tiene la posibilidad de generar logros de acuerdo a su progreso tanto en cinturones obtenidos como en posiciones obtenidas en distintos campeonatos.

28-29-30-31 Gestionar Wazas (añadir/editar/eliminar)

El alumno dispone de una herramienta para controlar sus *Wazas* (una combinación de 6 técnicas). La web permite obtener 6 fotos y montar un carrusel con ellas. Si hay alguna foto que no le gusta al alumno, puede borrarla y sacarla de nuevo. En caso de no estar contento con todo el resultado, puede eliminar el carrusel totalmente.

32 Subir Waza

El alumno haciendo uso de la web podrá subir el carrusel de fotos a una web de almacenamiento de fotos, mediante su *API* correspondiente.

33 Grabar entrenamiento

Se dará la posibilidad de grabar el entrenamiento a través de la web.

34 Subir entrenamiento

A través del uso de una *API*, el alumno podrá subir su entrenamiento a una página de almacenamiento de vídeos on-line.

35 Borrado de exámenes

El profesor podrá borrar los exámenes de un grupo entero desde su perfil.

36 Ver tabla alumnos

El profesor desde su perfil verá una tabla con todos los alumnos asociados a él mediante el código de profesor.

37 Ver sesión

El alumno podrá ver la sesión que ha grabado pinchando en el icono correspondiente en la página principal.

38/39 Notificar (sesión/waza)

El alumno podrá notificar al profesor que ha realizado una sesión y/o una waza.

40 Ver waza

El alumno podrá ver la waza que ha creado pinchando en el icono correspondiente en la página principal.

41 Marcar notificación

El profesor podrá marcar una notificación como vista para quitarla del panel principal.

4.3 Modelo de dominio

A continuación, se muestra el modelo de dominio asociado al TFG donde se pueden ver los tipos de datos que se almacenan.

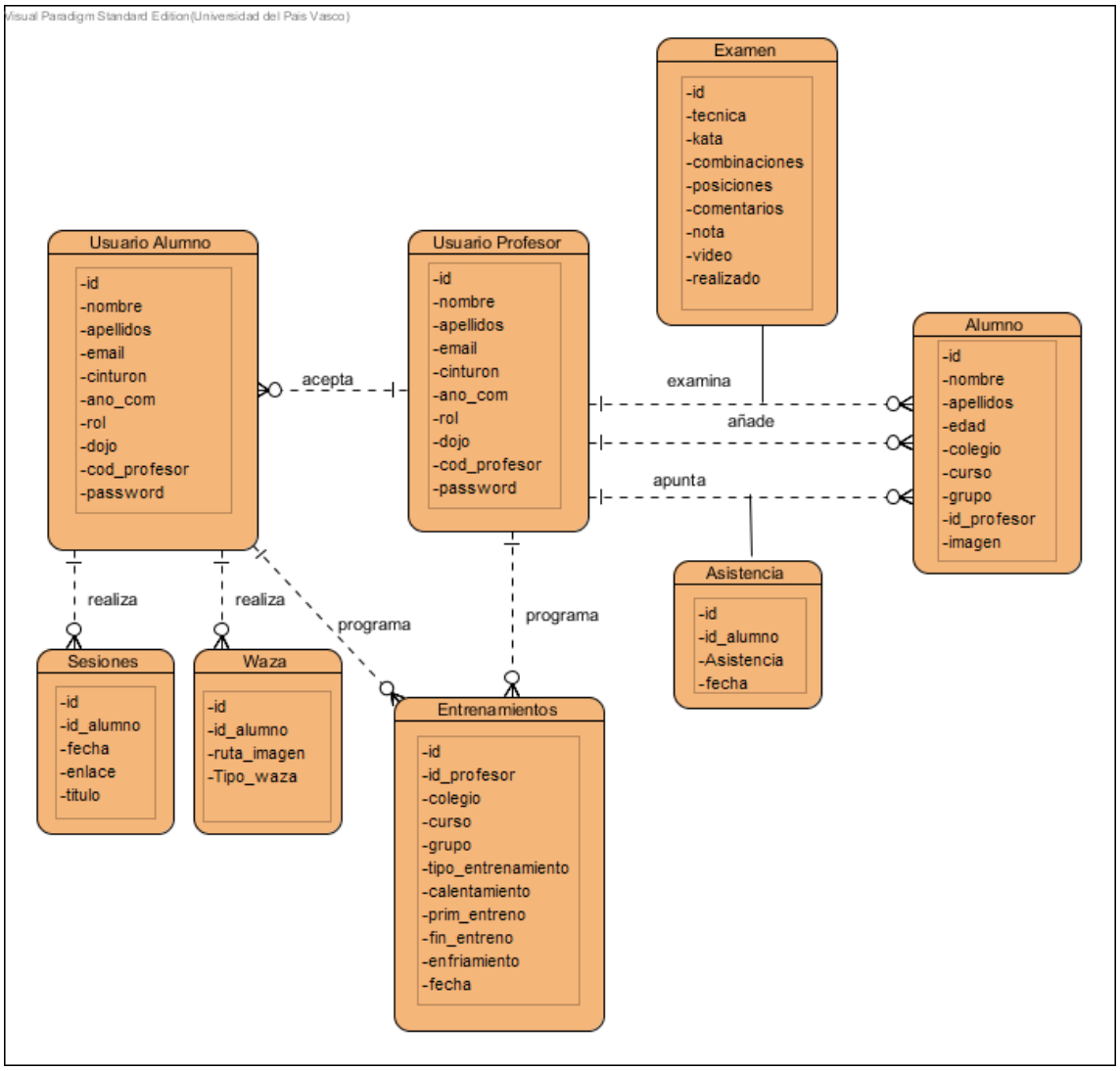


Ilustración 20 Modelo de dominio

En el modelo podemos ver las relaciones que hay entre las diferentes entidades que generan información a guardar.

Podemos ver que el usuario profesor añade varios alumnos y que por su parte un alumno es gestionado por un solo profesor. También pasa lista a los alumnos y les realiza exámenes.

El usuario alumno puede generar *Wazas* y sesiones. Tanto el profesor como el alumno pueden crear varios entrenamientos y los entrenamientos son generados por un profesor o un alumno.

5. Análisis y diseño

En este apartado se verán los aspectos imprescindibles para el desarrollo de la aplicación, se especificará el diseño de la base de datos y las tablas que las contienen para que sea más fácil su entendimiento y así poder contextualizarla junto al trabajo. También se detallará la estructura de la página web, las diferentes carpetas que componen todas las vistas, así como los controladores y el modelo que se encarga de unir la lógica del controlador con la base de datos.

5.1 Diseño de la base de datos

Las bases de datos se basan en un modelo relacional. Este modelo presenta tablas que contienen atributos en forma de filas y en ellas se encuentran los datos. Las tablas se relacionan entre sí para poder gestionar los datos de manera más eficiente. Para poder gestionar una base de datos relacional se usa el lenguaje SQL.

En la siguiente captura vemos un ejemplo de tabla en una base de datos. Vemos varias filas que representan la configuración del dato que van a alojar. Por ejemplo "ID" es un numero de longitud máxima 11(int (11)) no tiene atributos, no puede ser nulo, y es auto-incremental.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	1	ID	int(11)			No	*Ninguna*	AUTO_INCREMENT
☐	2	ID_profesor	int(11)			No	*Ninguna*	
☐	3	Colegio	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
☐	4	Curso	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
☐	5	Grupo	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
☐	6	Tipo_entrenamiento	varchar(50)	utf8_spanish_ci		No	*Ninguna*	
☐	7	Calentamiento	varchar(200)	utf8_spanish_ci		No	*Ninguna*	
☐	8	Prim_entreno	varchar(200)	utf8_spanish_ci		No	*Ninguna*	
☐	9	Fin_entreno	varchar(200)	utf8_spanish_ci		No	*Ninguna*	
☐	10	Enfriamiento	varchar(200)	utf8_spanish_ci		No	*Ninguna*	
☐	11	Fecha	varchar(20)	utf8_spanish_ci		No	*Ninguna*	

Ilustración 21 Esquema Bd

Para este proyecto se ha elegido *MySQL* como ya se ha citado en el apartado de herramientas, el cual nos proporciona todas las mecánicas que se necesitan para gestionar los datos.

Se ha elegido este sistema puesto que es gratis, se adapta a multitud de lenguajes de programación, es fiable, rápido y está ampliamente extendido por lo que buscar soluciones a problemas que pueden suceder es más sencillo.

Las tablas de la base de datos se han diseñado usando el programa *phpMyAdmin*, herramienta que facilita la misma empresa creadora del *MySQL*. Esta herramienta presenta una interface gráfica que resulta muy útil para crear/modificar/eliminar tablas o bases de datos completas de manera muy rápida.

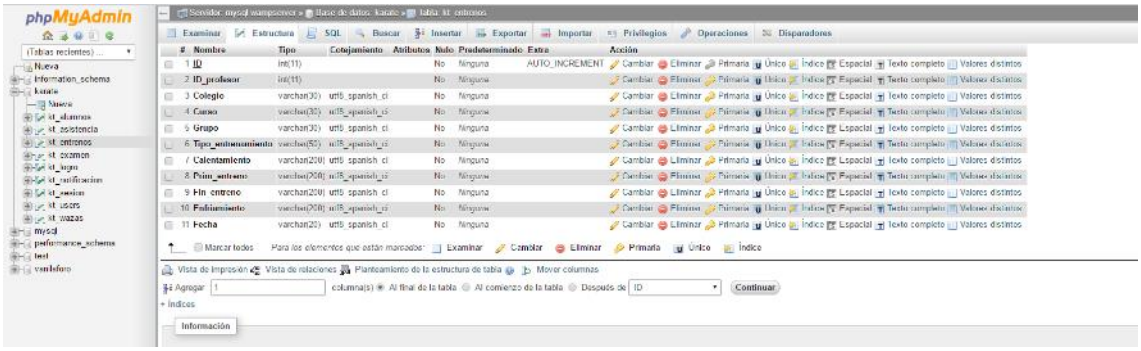


Ilustración 22Interface phpMyAdmin

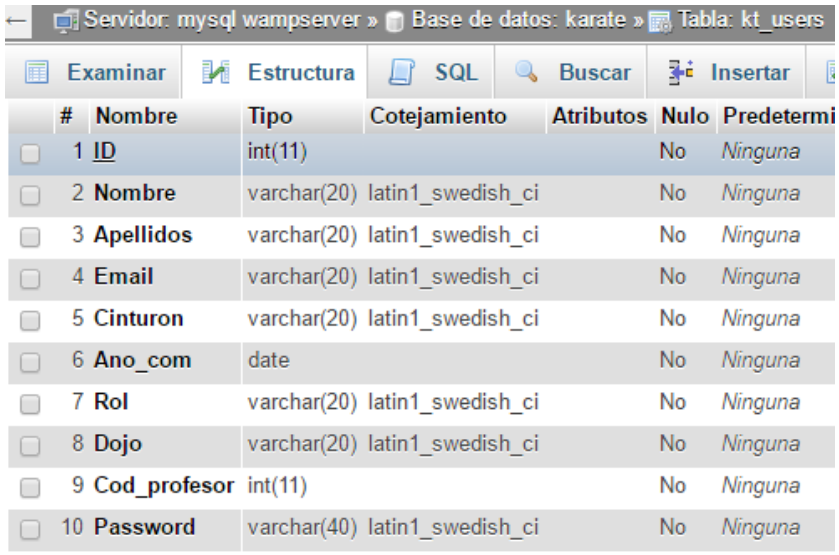
5.1.1 Modelo relacional y descripción detallada de la base de datos.

En este apartado veremos algunas tablas que han sido necesarias para el almacenamiento de los datos. Se mostrarán solo algunas para no extender demasiado la documentación con las mismas explicaciones.

La explicación de las restantes bases de datos se encuentra en el siguiente enlace:

https://dl.dropboxusercontent.com/u/13600207/proyecto/Mas%20bases%20de%20datos.pdf

5.1.1.1 Tabla de usuarios “kt_users”



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predetermi
1	ID	int(11)			No	Ninguna
2	Nombre	varchar(20)	latin1_swedish_ci		No	Ninguna
3	Apellidos	varchar(20)	latin1_swedish_ci		No	Ninguna
4	Email	varchar(20)	latin1_swedish_ci		No	Ninguna
5	Cinturon	varchar(20)	latin1_swedish_ci		No	Ninguna
6	Ano_com	date			No	Ninguna
7	Rol	varchar(20)	latin1_swedish_ci		No	Ninguna
8	Dojo	varchar(20)	latin1_swedish_ci		No	Ninguna
9	Cod_profesor	int(11)			No	Ninguna
10	Password	varchar(40)	latin1_swedish_ci		No	Ninguna

Ilustración 23 Kt_users

Esta tabla almacena a los usuarios que se registran en la web, pueden ser tanto profesores como alumnos. Estos datos se graban hacer el registro.

- ID → Guarda el ID de la persona que se registra, es auto incremental y se crea sin participación del usuario.
- Nombre → Guarda el nombre del usuario.
- Apellido → Guarda los apellidos del usuario.
- Email → Guarda el email del usuario.
- Cinturón → Guarda el cinturón de karate que tiene el usuario.
- Ano → Guarda el año de comienzo de la práctica del karate por parte del usuario...
- Dojo → Lugar donde practica karate el usuario.
- Cod_profesor → Este número es aleatorio y único, identificará al profesor. El sistema le asignará uno que deberá compartir con los alumnos si se quieren registrar y poder mandarle sesiones o wazas para que las vea. El alumno en poder de ese código lo tendrá que introducir en el registro.
- Password → Será el código de seguridad que use el usuario para entrar en el sistema, se codifica bajo MD5.

5.1.1.2 Tabla asistencia "kt_asistencia"

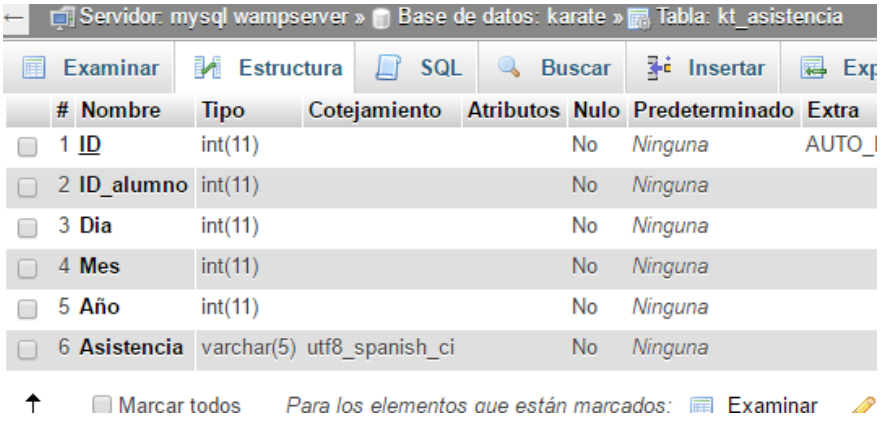


Ilustración 24 Tabla asistencia

En esta tabla se guarda la asistencia de los alumnos los días que se pasa en la web.

- ID → Guarda el ID, es auto incremental y se crea sin participación del usuario.
- ID_alumno → ID del alumno de la tabla "kt_alumnos"
- Día → Guarda el día en el que se pasa la asistencia.
- Mes → Guarda el mes en el que se pasa la asistencia.
- Año → Guarda el año en el que se pasa la asistencia.
- Asistencia → Guarda si ha venido o no el alumno a clase.

5.1.1.3 Tabla examen “kt_examen”

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	ID	int(11)			No	Ninguna
2	ID_alumno	int(11)			No	Ninguna
3	ID_profesor	int(11)			No	Ninguna
4	Tecnica	int(11)			No	Ninguna
5	Kata	int(11)			No	Ninguna
6	Combinaciones	int(11)			No	Ninguna
7	Posiciones	int(11)			No	Ninguna
8	Comentarios	varchar(200)	utf8_spanish_ci		No	Ninguna
9	Nota	decimal(11,0)			No	Ninguna
10	Video	varchar(200)	utf8_spanish_ci		No	Ninguna
11	Realizado	varchar(10)	utf8_spanish_ci		No	No
12	Fecha	varchar(20)	utf8_spanish_ci		No	Ninguna

Ilustración 25 Tabla examen

En esta tabla se guarda todo lo relacionado con el examen que realizan los alumnos.

- ID → Guarda el ID, es auto incremental y se crea sin participación del usuario.
- ID_alumno → ID del alumno de la tabla “kt_alumnos”
- ID_profesor → guarda la ID del profesor que los ha dado de alta, este ID es la misma que la ID de la tabla “kt_users”.
- Técnica → Guarda la nota obtenida en el apartado de técnica.
- Kata → Guarda la nota obtenida en el apartado de kata.
- Combinaciones → Guarda la nota obtenida en el apartado de combinaciones.
- Posiciones → Guarda la nota obtenida en el apartado de posiciones.
- Comentarios → Guarda los comentarios del profesor.
- Nota → Se calcula la nota obtenida y se guarda.
- Vídeo → Se guarda la dirección web donde está alojado el vídeo.
- Realizado → Se guarda si el examen ha sido realizado o no
- Fecha → Se guarda la fecha en la que realiza el examen.

Por último, se añade una ilustración con el esquema relacional de la base de datos al completo.

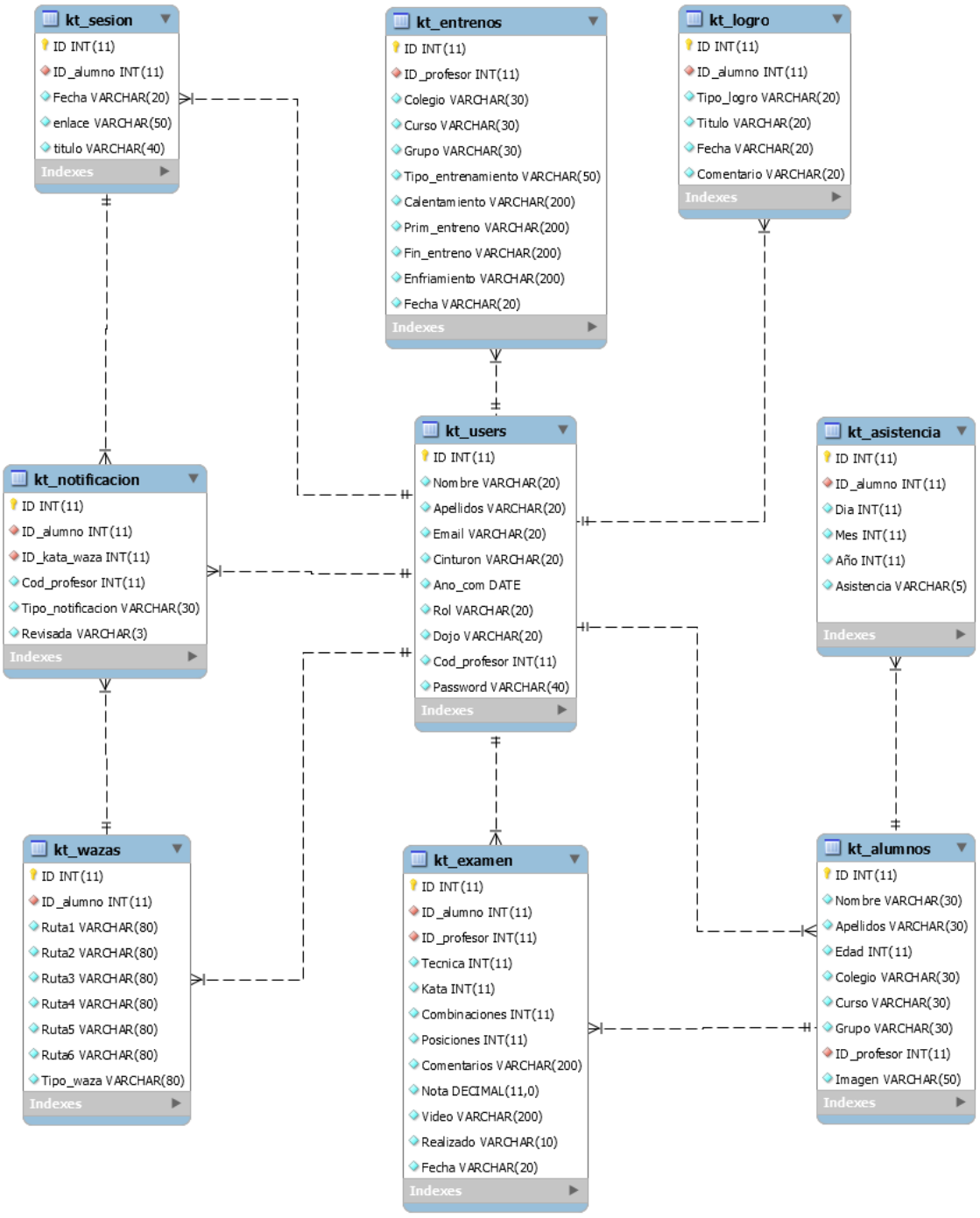


Ilustración 26 Esquema BBDD

5.2 Estructura de la aplicación web

En este apartado se buscar dar una visión general de la aplicación web, la estructura que sigue, los lenguajes utilizados, y el sistema de archivos seguidos para la consecución del proyecto.

Como se ha comentado anteriormente la web funcionará en todos los dispositivos que tengan internet adaptándose al tamaño de la pantalla de manera *Responsive.* La web también hace uso de varios *APIs* para dar funcionalidades extra. En este apartado se detallarán cuáles son las elegidas y como se incluyen en el proyecto.

5.2.1 Separación modelo-vista-controlador

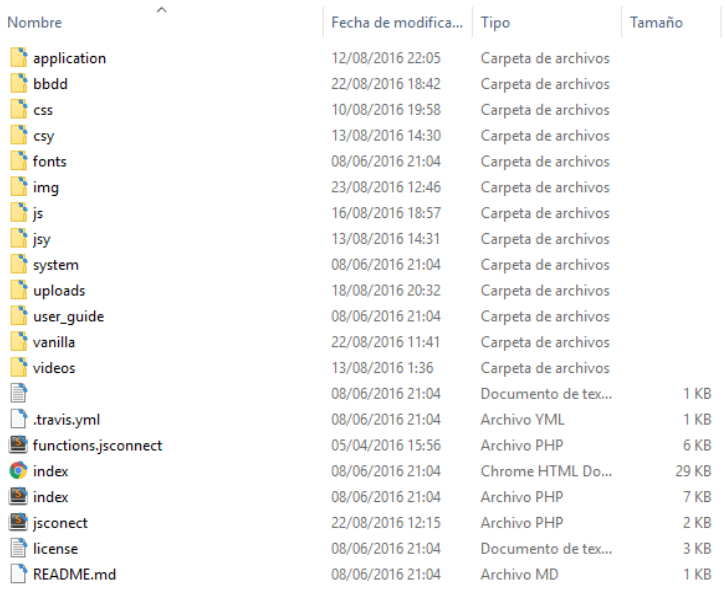
Gracias al uso del *framework CodeIgniter* se ha separado toda la arquitectura en tres partes; el modelo será encargado de la comunicación con la base de datos *MySQL* detallada en el punto anterior. La vista es la interfaz que ve el usuario y es la encargada de recoger los datos introducidos y mostrarlos. Por último, el controlador se encargará de la parte intermedia, recogerá los datos de la vista y los enviará a la base de datos si es necesario guardarlos y los procesará para enviárselos de nuevo para que los vea el usuario.

Las tecnologías usadas en la implementación de la web han sido las siguientes:

- Lenguaje PHP, se adapta perfectamente a las necesidades de la web y tiene herramientas suficientes y recursos para la construcción del proyecto.
- Hojas de estilos CSS, permite aplicar diferentes estilos a la web con el fin de que sea más atractiva para el usuario final. Mediante *Bootstrap* conseguimos acabados y animaciones que hacen de la web un sitio moderno y visualmente atractivo.
- Lenguaje JavaScript, nos permite mejorar el rendimiento de algunas funcionalidades.
- MySQL como sistema de gestión de la base de datos.

En el momento de la escritura de esta documentación el proyecto se carga en el servidor local del ordenador, proporcionado por *Wamp.* Se tiene previsto que cuando el proyecto este correctamente terminado y funcionando perfectamente se suba a un *hosting* para poder acceder como cualquier web normal. El servidor que se escoja debe soportar las tecnologías descritas previamente.

CodeIgniter, como hemos comentado anteriormente, separa en tres partes la estructura de la web creando un sistema de carpeta tal y como se muestra en la siguiente imagen.



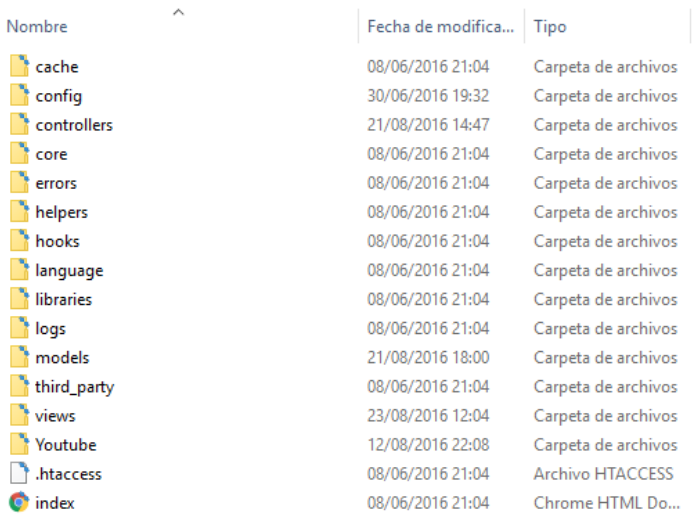
Nombre	Fecha de modifica...	Tipo	Tamaño
application	12/08/2016 22:05	Carpeta de archivos	
bbdd	22/08/2016 18:42	Carpeta de archivos	
css	10/08/2016 19:58	Carpeta de archivos	
csy	13/08/2016 14:30	Carpeta de archivos	
fonts	08/06/2016 21:04	Carpeta de archivos	
img	23/08/2016 12:46	Carpeta de archivos	
js	16/08/2016 18:57	Carpeta de archivos	
jsy	13/08/2016 14:31	Carpeta de archivos	
system	08/06/2016 21:04	Carpeta de archivos	
uploads	18/08/2016 20:32	Carpeta de archivos	
user_guide	08/06/2016 21:04	Carpeta de archivos	
vanilla	22/08/2016 11:41	Carpeta de archivos	
videos	13/08/2016 1:36	Carpeta de archivos	
	08/06/2016 21:04	Documento de tex...	1 KB
.travis.yml	08/06/2016 21:04	Archivo YML	1 KB
functions.jsconnect	05/04/2016 15:56	Archivo PHP	6 KB
index	08/06/2016 21:04	Chrome HTML Do...	29 KB
index	08/06/2016 21:04	Archivo PHP	7 KB
jsconect	22/08/2016 12:15	Archivo PHP	2 KB
license	08/06/2016 21:04	Documento de tex...	3 KB
README.md	08/06/2016 21:04	Archivo MD	1 KB

Ilustración 27 CodeIgniter 1

Una vez entramos en la carpeta del proyecto vemos la captura anterior. Como se ve en un primer vistazo se tienen todos los módulos separados. La mayor parte de estas carpetas las crea automáticamente la primera vez que instalamos el *framework*. Algunas otras han sido creadas por mí.

- Application: En esta carpeta es donde se encuentra toda la estructura que hemos creado de la web. En ella se encuentran tanto las vistas como los modelos y controladores. Esta carpeta la veremos mejor más adelante.
- Bbdd: en esta carpeta guardamos las copias de seguridad de la base de datos.
- CSS: aquí se guardan todas las hojas de estilos ".CSS" que necesitaremos para el diseño de la web.
- Csy: es una carpeta que guarda también ".CSS", pero esta vez es para la *API* de *YouTube* así que preferí dejarla separada del resto de CSS's.
- Img: guarda todas las imágenes que se usan para la web.
- Fonts: guarda todas las fuentes necesarias para los textos de la web.
- Js: en esta carpeta se guardan todos los ".js" que son archivos JavaScript necesarios para varias funcionalidades de la web.
- Jsy: mismo caso que en la carpeta csy, aquí se guardan los ".js" referentes al *API* de *YouTube*.
- Uploads: carpeta donde se suben todas las imágenes que suban los usuarios.
- Vanilla: aquí se almacena la *API* de *vanillaforums* que es el encargado de crear el foro en la web.

A continuación, se muestra una captura de la carpeta "applications" la cual contiene el modelo-vista-controlador que explicaremos después.

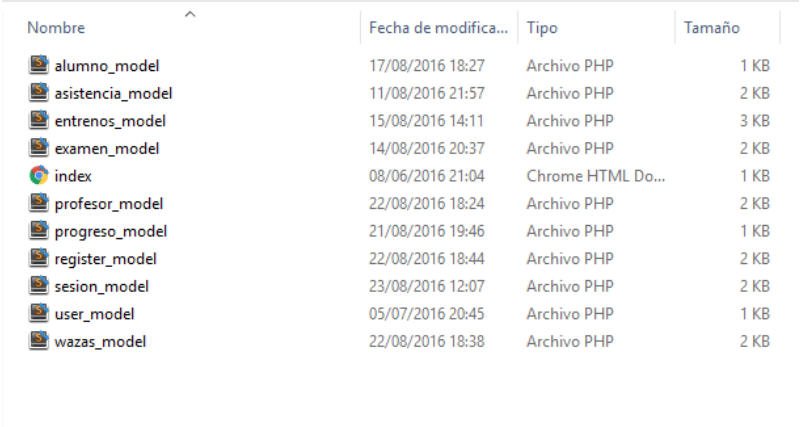


Nombre	Fecha de modifica...	Tipo
cache	08/06/2016 21:04	Carpeta de archivos
config	30/06/2016 19:32	Carpeta de archivos
controllers	21/08/2016 14:47	Carpeta de archivos
core	08/06/2016 21:04	Carpeta de archivos
errors	08/06/2016 21:04	Carpeta de archivos
helpers	08/06/2016 21:04	Carpeta de archivos
hooks	08/06/2016 21:04	Carpeta de archivos
language	08/06/2016 21:04	Carpeta de archivos
libraries	08/06/2016 21:04	Carpeta de archivos
logs	08/06/2016 21:04	Carpeta de archivos
models	21/08/2016 18:00	Carpeta de archivos
third_party	08/06/2016 21:04	Carpeta de archivos
views	23/08/2016 12:04	Carpeta de archivos
Youtube	12/08/2016 22:08	Carpeta de archivos
.htaccess	08/06/2016 21:04	Archivo HTACCESS
index	08/06/2016 21:04	Chrome HTML Do...

Ilustración 28 Carpeta applications

5.2.1.1 Modelo

Corresponde a la carpeta *"models"* en ella tenemos la siguiente vista.



Nombre	Fecha de modifica...	Tipo	Tamaño
alumno_model	17/08/2016 18:27	Archivo PHP	1 KB
asistencia_model	11/08/2016 21:57	Archivo PHP	2 KB
entrenos_model	15/08/2016 14:11	Archivo PHP	3 KB
examen_model	14/08/2016 20:37	Archivo PHP	2 KB
index	08/06/2016 21:04	Chrome HTML Do...	1 KB
profesor_model	22/08/2016 18:24	Archivo PHP	2 KB
progreso_model	21/08/2016 19:46	Archivo PHP	1 KB
register_model	22/08/2016 18:44	Archivo PHP	2 KB
sesion_model	23/08/2016 12:07	Archivo PHP	2 KB
user_model	05/07/2016 20:45	Archivo PHP	1 KB
wazas_model	22/08/2016 18:38	Archivo PHP	2 KB

Ilustración 29 Carpeta Models

Dentro vemos varios archivos PHP. Cada cual contiene el código que hace la transmisión de datos entre el controlador y la base datos y viceversa. Usaremos estos archivos para guardar o sacar información de la base de datos.

Se pueden crear tantos archivos como se necesite, aplicándoles un nombre identificativo para el trabajo que realizan. Por ejemplo "profesor_model.php" se encarga de todos los datos relacionados con el profesor. Esto viene bien para tenerlos separados por usuarios o funcionalidades y que sea más fácil agregar entradas a estos archivos.

Una vista rápida de lo que contiene un archivo del modelo es la siguiente.

```php
function datos_profesor($ID){
    $sql = "SELECT * FROM `kt_users` WHERE `ID` = $ID ";
    $resultado=$this->db->query($sql)->result();
    return $resultado;
}
```

Ilustración 30 Profesor_model

En ella podemos ver una llamada a la base de datos para recoger todos los datos del profesor según el ID que reciba. El *return* mandará los datos de vuelta al controlador para realizar lo necesario con ellos.

5.2.1.2 Vista

Representa la carpeta *"views"* de la ilustración 28. En ella se guardan todas las vistas que verá el usuario en la pantalla. Sirven como entrada y salida de datos. Tenemos la siguiente vista de archivos PHP.

Nombre	Fecha de modifica...	Tipo	Tamaño
alumno	22/08/2016 16:53	Archivo PHP	19 KB
asistencia	12/07/2016 18:02	Archivo PHP	10 KB
clase	12/07/2016 23:01	Archivo PHP	9 KB
colegial	26/07/2016 20:22	Archivo PHP	11 KB
crear_geriw	18/08/2016 21:20	Archivo PHP	12 KB
crear_sukiw	18/08/2016 20:43	Archivo PHP	12 KB
crear_uchiw	18/08/2016 21:22	Archivo PHP	12 KB
crear_ukew	18/08/2016 21:22	Archivo PHP	12 KB
edit_colegial	12/07/2016 19:58	Archivo PHP	12 KB
edit_entreno	11/08/2016 20:45	Archivo PHP	13 KB
edit_entreno_alumno	15/08/2016 14:08	Archivo PHP	12 KB
entrenamientos	10/08/2016 21:18	Archivo PHP	10 KB
examinado	13/08/2016 17:42	Archivo PHP	16 KB
grabar_examen	13/08/2016 15:02	Archivo PHP	11 KB
grabar_sesion	16/08/2016 20:26	Archivo PHP	11 KB
logros	21/08/2016 18:49	Archivo PHP	10 KB
main	23/08/2016 12:52	Archivo PHP	39 KB
new_alumno	17/08/2016 18:45	Archivo PHP	11 KB
new_belt	21/08/2016 19:39	Archivo PHP	10 KB
new_comp	21/08/2016 19:39	Archivo PHP	10 KB

Ilustración 31 Carpeta views

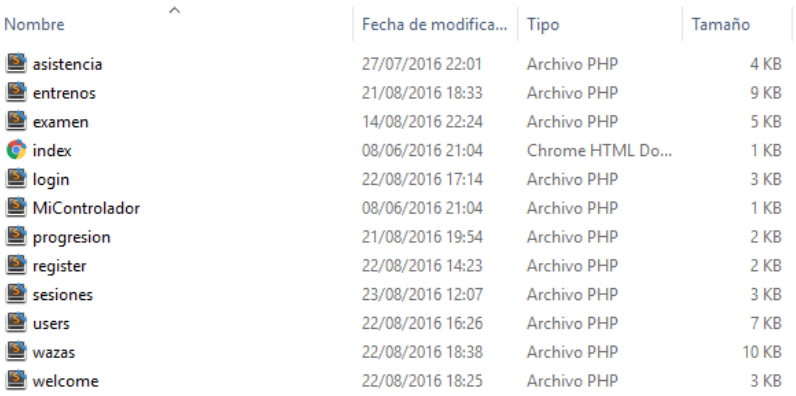
En esta carpeta se encuentran muchos más archivos de los mostrados, tantos como pantallas tenga la web al completo. Al igual que en el modelo se pueden crear tantos como se requiera, que suele ser un número muy elevado y superior al modelo y al controlador, puesto que estos dos concentran las funcionalidades.

En las vistas conviven el lenguaje PHP para los datos a mostrar, el lenguaje HTML5 para las etiquetas y el lenguaje JavaScript para algunas funcionalidades.

Dado que la vista interna de estos archivos no difiere en exceso de un archivo PHP o un HTML, no se mostrará una captura interna del archivo.

5.2.1.3 Controlador

Corresponde a la carpeta *"controlers"* de la ilustración 28. En ella encontramos la siguiente captura.



Nombre	Fecha de modifica...	Tipo	Tamaño
asistencia	27/07/2016 22:01	Archivo PHP	4 KB
entrenos	21/08/2016 18:33	Archivo PHP	9 KB
examen	14/08/2016 22:24	Archivo PHP	5 KB
index	08/06/2016 21:04	Chrome HTML Do...	1 KB
login	22/08/2016 17:14	Archivo PHP	3 KB
MiControlador	08/06/2016 21:04	Archivo PHP	1 KB
progresion	21/08/2016 19:54	Archivo PHP	2 KB
register	22/08/2016 14:23	Archivo PHP	2 KB
sesiones	23/08/2016 12:07	Archivo PHP	3 KB
users	22/08/2016 16:26	Archivo PHP	7 KB
wazas	22/08/2016 18:38	Archivo PHP	10 KB
welcome	22/08/2016 18:25	Archivo PHP	3 KB

Ilustración 32 Carpeta controllers

Presenta una visión muy similar al modelo, ya que como hemos comentado tiende a concentrar todas las funcionalidades en un archivo. Por ejemplo, el archivo "entrenos" guarda todo lo relacionado con los entrenamientos para que sea más fácil localizar un error o ampliar una funcionalidad, teniéndolo todo ordenado.

Los controladores se encargan de la lógica de la página, cogen los datos de la vista y trabajan con ellos. Si es necesario cargar o guardar algún dato se comunican con el modelo correspondiente y posteriormente vuelven a la vista para mostrar el resultado.

A continuación, se muestra una pequeña parte de cómo se compone un archivo controlador.

```php
class Progresion extends CI_Controller {

  function crear_logro(){

    $this->load->view('logros');
  }

  function elegir_logro($tipo){

    switch ($tipo) {
      case '1':
        $this->load->view('new_comp');
        break;
      case '2':
        $this->load->view('new_belt');
        break;

      default:
        redirect('../index.php?/crear_logro', 'refresh');
        break;
    }

  }
```

Ilustración 33 Archivo controlador

En ella se ven dos funciones; la primera, "crear_logro" nos redirecciona a la vista "logros" que muestra por pantalla los logros obtenidos por el usuario.

En la segunda vemos “elegir_logro”, la cual es llamada desde una vista y según lo elegido por el usuario carga una vista u otra.

5.2.2 *API*

Las *APIs* nos proporcionan conexiones con diversas funcionalidades de otras páginas de una manera sencilla. Por ejemplo, en la web se usa la subida de vídeos mediante la *API* de *YouTube*, que nos permite alojar el vídeo en sus servidores.

En este apartado veremos qué *APIs* hemos escogido para cada tarea y por qué. En la sección de desarrollo (6) veremos cómo se han implementado, los problemas que han surgido y cómo se han atajado.

5.2.2.1 *API* de *YouTube*

Cuando se usa desde un móvil la web graba a través de la cámara del dispositivo. En el ordenador, sin embargo, se ha de seleccionar un archivo previamente grabado. Una vez se dispone del archivo es necesario subirlo a una plataforma de almacenamiento de vídeos.

Para ello se barajaron diferentes alternativas, entre ellas *Vimeo, YouTube, Mega…*De ellas se tuvo en cuenta la *API* y si era fácil su integración mediante PHP o JavaScript. También se tuvo en cuenta su centro de control y la petición de permisos de subida. Por ejemplo, para *Google* y su *YouTube* la petición de permiso para poder subir vídeos es instantánea tras configurar un par de apartados. *Vimeo* y *Mega* necesitan que se rellene un cuestionario y que su equipo lo valide, una petición que se puede demorar hasta siete días según sus políticas.

Se tuvo en cuenta la documentación que presentaban desde el apartado “desarrollador” para ayudar a implementar la *API*. *YouTube* presenta una documentación muy cuidada disponible en multitud de lenguajes de programación, mientras que en sus competidoras la documentación era más confusa. Para terminar *YouTube* tenía más presencia en la red en cuanto a ayudas por parte de usuarios y ejemplos de subidas.

Por estas razones se escogió *YouTube* como plataforma de subida de vídeos online.

La integración del proyecto se realiza mediante JavaScript y CSS. Como vimos en el apartado anterior cuentan con dos carpetas “jsy” y “csy” que contienen los archivos necesarios para la subida. Este apartado se verá con mayor detalle en la siguiente sección.

5.2.2.2 *API de Google Drive*

En el apartado de *Wazas* el alumno saca 6 fotos y se crea un carrusel. Las fotos que sean seleccionadas se podrán subir a un servicio de almacenamiento en la nube.

En este apartado también se barajaron diferentes opciones como *Dropbox, Google Drive, Mega*….

Como ya se ha mencionado, *Mega* tiene en su política para desarrolladores un tiempo de espera de respuesta de siete días, la cual hace que no sea una opción a tener en cuenta.

La opción principal fue usar *Dropbox* puesto que es el servicio de almacenamiento que más uso y por tanto mejor conozco. Se descargó la *API* y se observó que presentaba muchas dificultades para su correcta implementación. Por tanto, se buscó ayuda en comunidades como *StackOverflow* en busca de ejemplos de implementación, pero los encontrados no presentaban una fácil modificación para adaptarlos al proyecto.

Se optó entonces por *Google Drive* después de los buenos resultados obtenidos y la facilidad de implementación de la *API* de *YouTube* en el proyecto. Esta *API* también presentó dificultad de inserción en el trabajo al igual que *Dropbox*. Tras muchas búsquedas y pruebas se encontró una solución que, aunque no es la más acertada, permite subir los archivos a *Google Drive*. Esta solución pasa por subir las fotos una a una en vez de todas a la vez, como se hubiera preferido en primera instancia. Al igual que la *API* de *YouTube*, se detallará mejor su implementación en el siguiente apartado.

5.2.2.3 *API* de *VanillaForum*

En los requisitos se establece que se necesita un punto en común para compartir ideas y comunicación. Por esta razón se implementa un foro.

Debido a la experiencia previa con la *API* de *Vanillaforum* se escogió esta opción. Su implementación es fácil y sencilla, con integración en la web mediante JavaScript.

5.2.2.4 *API* de *Google Chart*

En un principio se pensó incluir unas gráficas donde el usuario pudiera ver su progresión. Después de implementarlas y ver su funcionamiento se vio que no daba valor al usuario.

Se observó que las configuraciones de las gráficas no hacían fácil su lectura. Por ejemplo, en una gráfica de progresión, el número uno por defecto está en la parte más baja del eje vertical y en este caso se quiere colocar en el punto más alto para representar la mejor posición obtenida en un campeonato. También es difícil añadir y ampliar las gráficas con diferentes tipos de campeonatos puesto que muchos se realizan en el mismo año y no queda bien representado.

Por estas razones se decidió quitar la integración con *Google Chart* del proyecto.

6. Desarrollo

En el apartado anterior se han enumerado las tecnologías usadas, pero sin entrar en excesivo detalle.

En este apartado se detallará la implementación de cada funcionalidad. Se mostrarán líneas de código de ciertos archivos para ver el funcionamiento del modelo, la vista y el controlador y cómo se conectan entre ellos. También se especificarán con detalle el diseño y la implementación de las interfaces de usuario. Por último, se verá cómo se integran las *APIs* dentro del sistema, qué archivos necesitan y qué resultado producen.

Se incluirán algunas capturas; sobre todo las más relevantes o interesantes para no alargar la documentación.

El primer apartado que viene a continuación es la instalación previa donde se verá de forma muy resumida cómo se instala *CodeIgniter* y los primeros pasos a la hora de comenzar con el trabajo. Más tarde se verá el diseño de la web, un pequeño vistazo a las interfaces más relevantes, así como una descripción de su uso. Para finalizar volveremos a repasar toda la web, pero esta vez desde el punto de vista del código, mostrando las líneas de código más relevantes, así como interesantes. En este apartado se verá cómo se implementan las *APIs*.

Recordar por último que a la hora de realizar esta parte de la documentación el proyecto se encuentra en fase de mejora, tanto visual como de código. Las funcionalidades ya están terminadas, pero es posible que el apartado visual cambie ligeramente (botones, iconos o títulos) en la versión final.

6.1 Instalación previa.

El primer paso es descargar el *framework* de *CodeIgniter*. Una vez bajado, lo instalamos en el servidor. Para ello simplemente es necesario descomprimir el ".rar" del programa.

También es necesario tener instalado *Wamp* para emular un servidor web de manera local, la instalación de este programa es una instalación normal.

Una vez descargados ambos paquetes podemos empezar a programar el trabajo.

6.2 Diseño de la web

En este apartado se mostrarán las diferentes interfaces que componen el trabajo con diferentes capturas y una explicación de su funcionamiento. En la parte final del apartado se mostrarán las mismas interfaces, pero a través de un dispositivo móvil puesto que el trabajo está más pensado para ser usado desde una *tablet* que desde un ordenador. El diseño solo se realiza una vez ya que siendo *Responsive* permite adaptar la interface a distintas pantallas de manera automática.

Las interfaces como ya hemos comentado, se usan para capturar y mostrar datos. Están programadas en HTML5 puesto que su sistema de etiquetas es el más moderno y permite realizar muchas funcionalidades de manera simple y rápida.

Tras la instalación, empezamos el diseño de la web principal. Está creada con *Bootstrap* para darle un diseño más moderno.

El menú superior, visible en todo momento, permite la navegación a los diferentes apartados de la web.

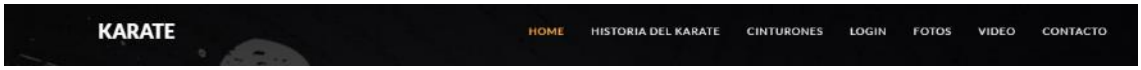


Ilustración 34 Menú

Al estar confeccionada con *Bootstrap*, la longitud de la página es muy grande y sacar una captura de toda la haría ilegible. Por ello vamos a ver cada sección mencionada anteriormente con una pequeña captura y descripción.

6.2.1 Página principal

Esta página es la primera que verá cualquier visitante que acceda a la web. Se muestran unas pequeñas explicaciones de los dos roles que puede asumir el usuario y las funcionalidades que tendrá en uno u otro rol.

6.2.2 Registro de nuevos usuarios

El usuario una vez elegido el rol, se podrá registrar rellenando el formulario correspondiente.

Ilustración 35 Registro

En esta pantalla el usuario deberá introducir su nombre, apellido, correo electrónico, contraseña, cinturón, año de comienzo y lugar de entrenamiento. La única diferencia entre

ambos roles será el campo “código profesor” que el profesor tendrá que facilitar al alumno para el registro.

6.2.3 Página usuario profesor

Una vez identificado como profesor la interface cambia para adaptarse a las funcionalidades del rol profesor.

El menú superior cambia mostrando: gestionar alumnos, gestionar entrenamientos y foro en la parte izquierda. En la parte derecha se encuentra el acceso al perfil del profesor y el logout para cerrar la sesión.

6.2.3.1 Notificaciones

El sistema permite al usuario alumno mandar notificaciones al profesor. Éstas pueden ser de dos tipos: *Wazas* o sesiones.

Las *Wazas*, como ya hemos mencionado anteriormente, es una combinación de 6 técnicas que serán fotografiadas. Una vez elegidas las fotos, la web creará un carrusel con ellas. El alumno podrá notificar al profesor para que éste eche un vistazo al carrusel. También podrá subir sus fotos favoritas a su página de almacenamiento en *Google Drive*.

Las sesiones son entrenamientos grabados y subidos a *YouTube*.

Una vez el alumno pulsa sobre notificar en una *Waza* o sesión, se le envía una alerta visual al profesor. Esta alerta se muestra en la parte superior de la pantalla y permanecerá ahí hasta que el profesor la marque como vista.

El apartado que contiene la notificación solo será visible si hay una notificación que mostrar, estando oculto en otro caso.

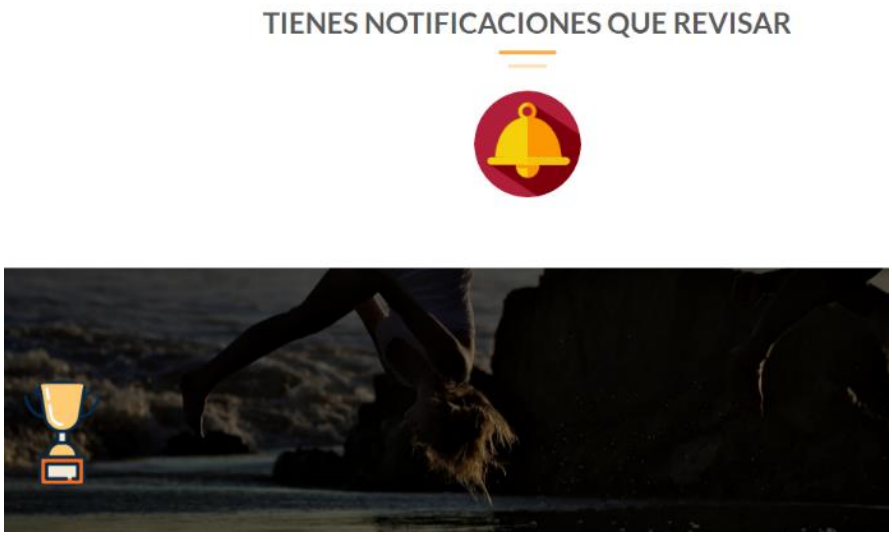


Ilustración 36 Notificaciones

6.2.3.2 Gestionar alumnos

En este apartado se le presenta al profesor una tabla de los alumnos que previamente ha introducido en el sistema. Conviene recordar que los alumnos que se registran ellos mismos no son los mismos que los que el profesor da de alta. Los primeros son alumnos que tienen acceso al área "alumnos" de la web y pueden hacer uso de todas las funcionalidades. A los que el profesor da de alta mediante el registro de alumnos no acceden a la web.

En el ejemplo siguiente ya hay alumnos (de pruebas) en el sistema, presentando la siguiente tabla:

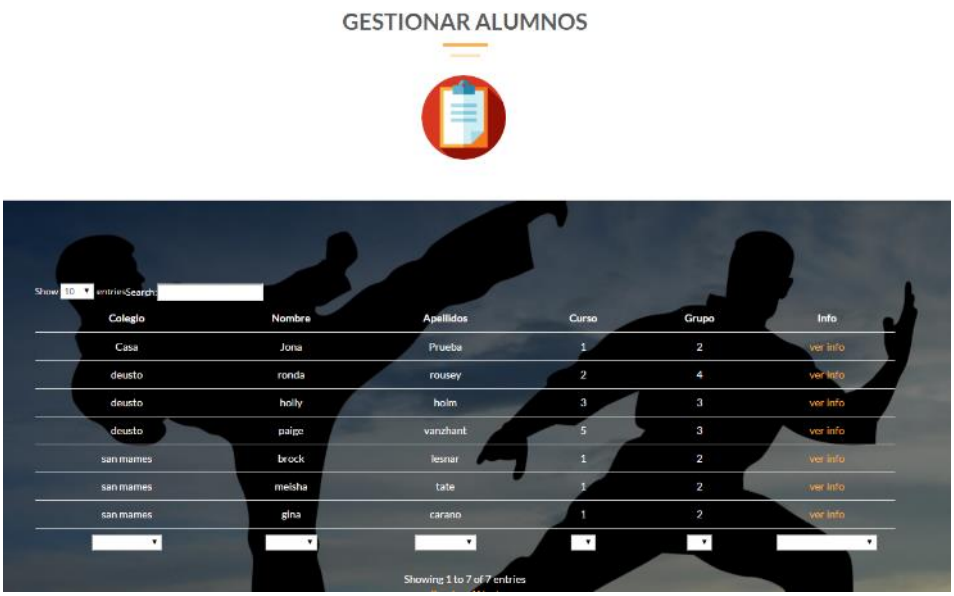


Ilustración 37 Gestionar alumnos

En ella se ven ordenados por colegio todos los alumnos (en caso de tener multitud de alumnos se crearían más páginas). Las casillas debajo de cada columna permiten filtrar la tabla, pudiendo filtrarse de manera combinada.

Al final de cada fila se ve "ver info"; la cual nos llevaría a la pantalla del alumno, donde se nos muestran todos sus datos y nos permite la edición de los mismos.

6.2.3.2.1 Nuevo alumno

En este apartado también se encuentra el botón "Nuevo alumno", el cual nos permite dar de alta un alumno nuevo, rellenando sus campos. Aquí nos encontramos con un requerimiento funcional, poder sacar una foto al alumno para rellenar la ficha.

En la siguiente imagen se muestra el formulario de alta de alumno, en el lado derecho se puede seleccionar una foto (botón "browse") para que se integre en su ficha. Como está

siendo documentada en el ordenador este botón permite cargar una foto desde cualquier carpeta. En cambio, sí pulsamos desde el móvil se nos abrirá automáticamente la cámara de fotos.

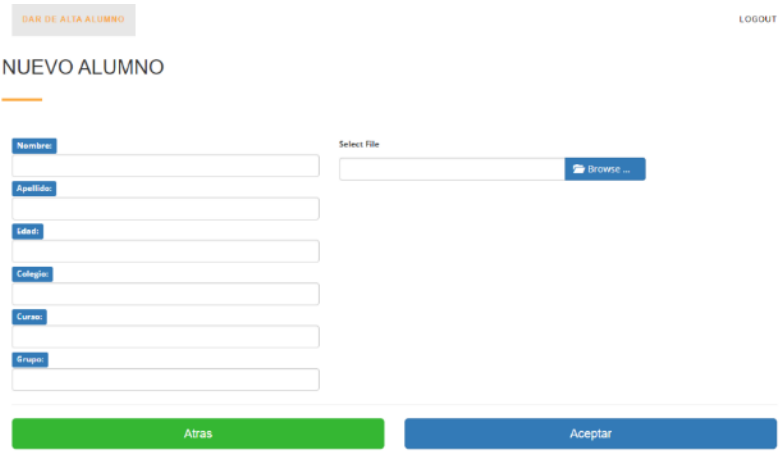


Ilustración 38 Dar de alta alumno

6.2.3.2.2 Gestionar alumnos

En la pantalla principal, debajo de la tabla y junto al botón de "nuevo alumno", encontramos el botón de "gestión de alumnos". Éste botón nos lleva a nuevas funcionalidades que son requisitos para el proyecto.

Una vez pulsado el botón nos muestra un desplegable donde podemos elegir el colegio. Después de elegirlo se nos mostrarán los distintos cursos y grupos donde hay alumnos registrados. En este apartado aparecen cuatro nuevos botones como en el ejemplo siguiente.

Ilustración 39 Gestión de alumnos

El botón "ver clase" nos permitirá obtener una pantalla con la foto de todos los alumnos que componen la clase.

El botón "control de asistencia" realiza una funcionalidad requerida para el trabajo. Se mostrarán las fotos de los alumnos en orden y se irá deslizando con el dedo a derecha o izquierda para marcar si el alumno está o no está presente. En el ordenador se realizará manteniendo pulsado el ratón sobre la foto y arrastrando hacia el lado elegido. La interface en estos momentos solo muestra la foto del alumno.

En “ver tabla asistencia” veremos una tabla con todos los alumnos marcando el día que se le paso lista. Si el alumno estuvo en clase se marcará con el color verde, en caso contrario se marcará de color rojo.

En el último botón tenemos “iniciar examen”; el cual nos traslada a una nueva interfaz donde se nos muestra una tabla. En ella vemos la lista de alumnos y si han realizado el examen. Si lo han hecho se mostrará la fecha en la que lo realizaron y la nota obtenida.

Ilustración 40 Tabla examen

En el caso anterior ninguno de los usuarios ejemplo ha hecho el examen. Si pulsamos sobre “iniciar examen” cargará el primer alumno, en este caso “brock lesnar” y mostrará la siguiente pantalla.

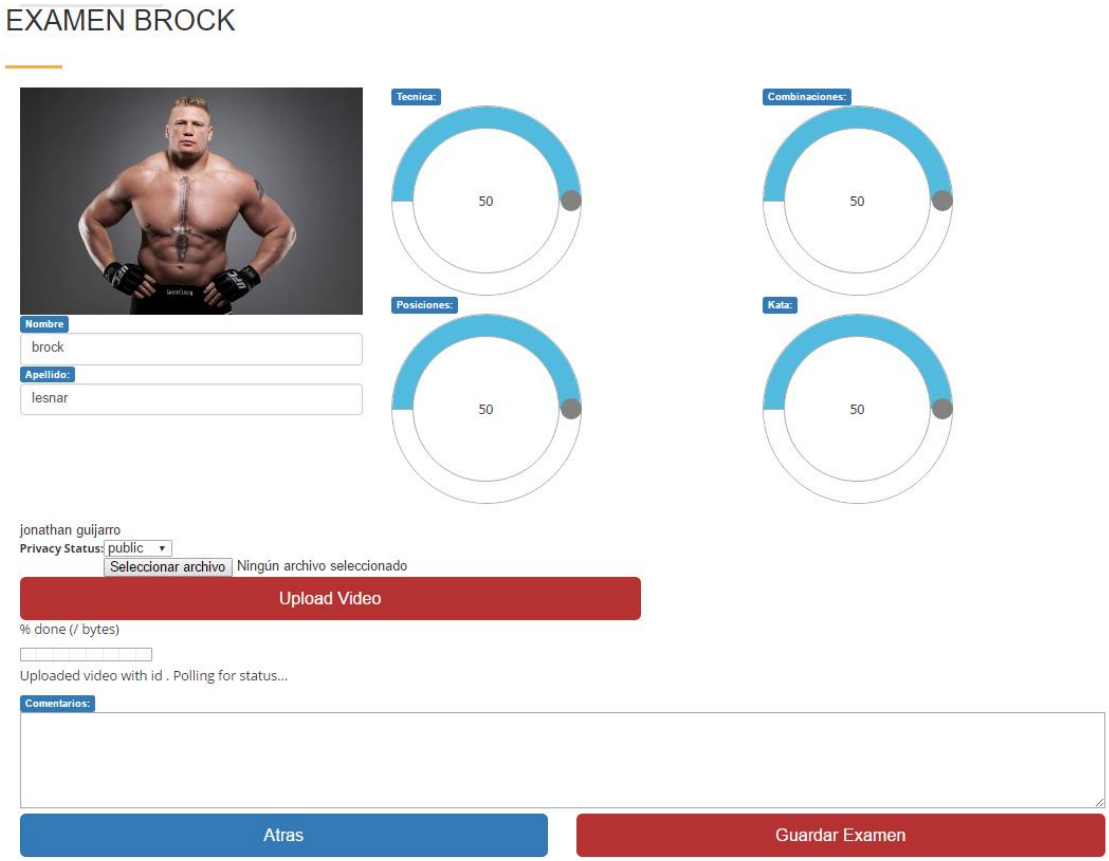


Ilustración 41 Pantalla examen

En esta pantalla vemos la foto del alumno junto con su nombre y apellido. En el lado derecho vemos 4 círculos; los cuales representan los valores a puntuar en el examen: técnica, combinaciones, posiciones y kata. Por defecto se presentan con un 50 que el

profesor podrá modificar para adaptarlo a la nota sobre 100 que crea que tiene el alumno en esa categoría.

Más abajo vemos un "seleccionar archivo"; el cual permite seleccionar un fichero de tipo vídeo para subirlo a *YouTube*. Como en apartados anteriores, al estar en un ordenador permite seleccionar un vídeo ya existente. Si se abre desde el móvil, dará acceso a la cámara y permitirá al profesor grabar el examen.

Una vez seleccionado el archivo de vídeo, bien por la ubicación o bien por la cámara, se dispone del botón "upload vídeo". Éste subirá el vídeo al servidor de *YouTube* mostrando un progreso en la barra inferior al botón. El sistema dará el enlace que se podrá usar posteriormente para ver el vídeo.

Más abajo tenemos el cajón de texto "comentarios" que el profesor podrá rellenar con comentarios sobre el examen.

Una vez este todo rellenado, pulsando sobre "guardar examen", el examen será almacenado y se mostrará al siguiente alumno.

Nada más terminar y guardar un examen la tabla anterior cambiará, mostrando la fecha y la nota obtenida. En caso de no finalizar una tanda de exámenes con todos los alumnos el proceso se podrá continuar en cualquier momento desde el alumno donde se dejó.

6.2.3.3 Gestión entrenamientos

De vuelta a la pantalla principal del profesor tenemos como siguiente apartado "gestionar entrenamientos" en el cual el profesor podrá crear un entrenamiento para una clase. Para ello en un desplegable similar al apartado anterior seleccionará el colegio y posteriormente el curso y el grupo. Desde esa pantalla podrá crear un entrenamiento nuevo o ver todos los entrenamientos que tiene ese grupo en concreto.

Pulsando sobre "añadir entrenamiento" el profesor podrá especificar qué tipo de entrenamiento se va a realizar. A continuación, rellenará los demás campos con los ejercicios escogidos para cada uno.

Una vez rellenados todos los campos, se volverá a la pantalla principal pero esta vez se añadirá un icono que le recordará el entrenamiento. Pinchando en el podrá ver los detalles de ese entrenamiento en concreto.

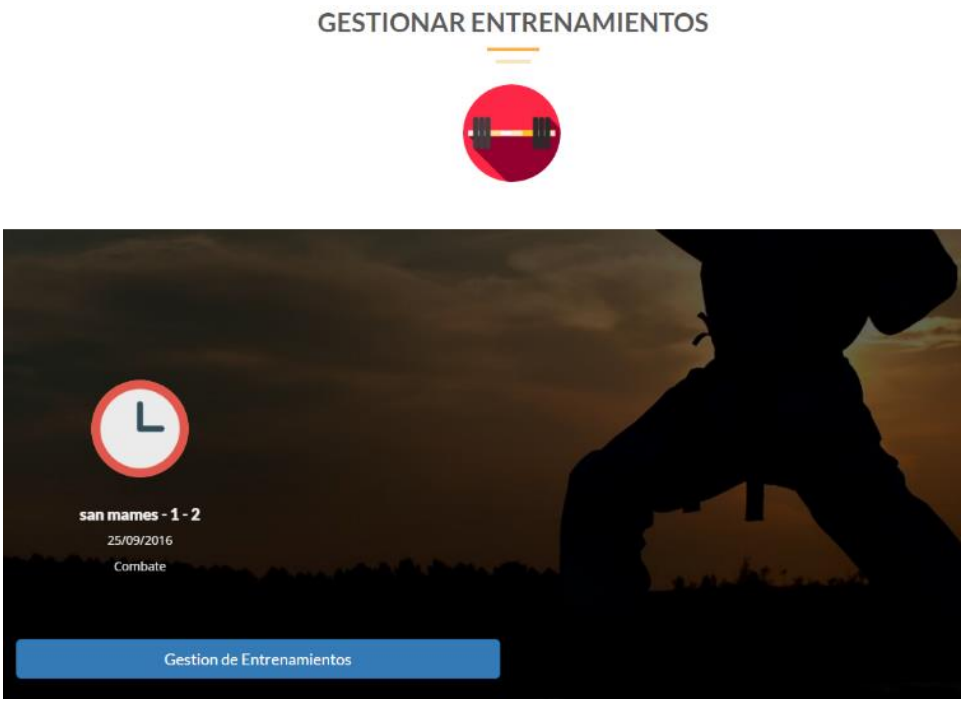


Ilustración 42 Gestionar entrenamiento

En la imagen podemos ver el icono que se crea sobre un texto que describe a qué colegio, curso y grupo está asociado el entrenamiento. Abajo tenemos la fecha en la que se realizará el entrenamiento, así como el tipo de entrenamiento que es.

6.2.3.4 Foro

Por último, en la pantalla principal del profesor se muestra el foro que permite una comunicación con el resto de usuarios.

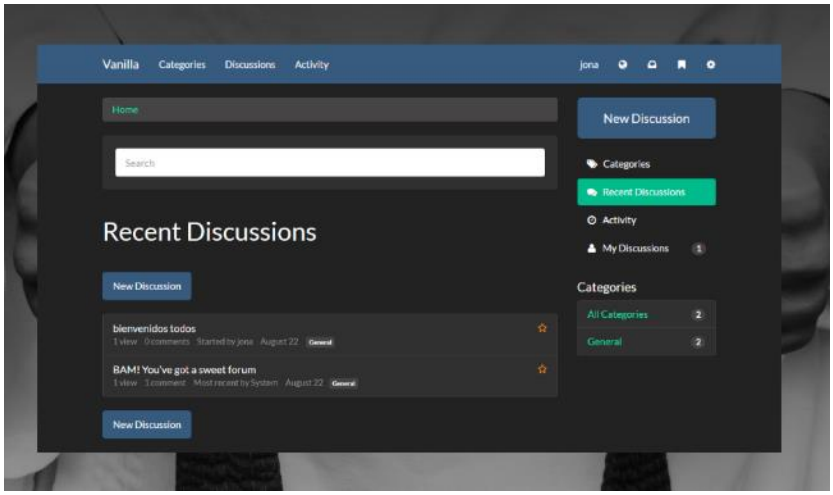


Ilustración 43 Foro

6.2.3.5 Perfil

En este apartado el profesor puede cambiar sus datos, así como su contraseña. También puede borrar los datos de los exámenes para poder realizar otra tanda sobre la misma clase. Esto es así debido a que un alumno no puede hacer dos veces el examen antes de que el resto lo haya hecho al menos una vez.

Con esta herramienta puede borrar todos los exámenes de una clase para volver a realizarlos.

Ilustración 44 Borrado de exámenes

En otro apartado el profesor verá a todos los usuarios alumnos a los que les ha dado su código de profesor y están asociados a él.

Ilustración 45 Lista de alumnos

6.2.4 Página usuario alumno

En este apartado se verán las interfaces que componen la pantalla del usuario alumno. Algunos apartados son similares a los del profesor y no se detallarán.

6.2.4.1 Progresiones y logros

El primer apartado que se muestra en el usuario alumno son los logros y progresiones. Este apartado permite al usuario introducir la fecha en la que participó en un torneo y la posición obtenida, así como marcar cuándo consiguió un cinturón y cuál fue.

Una vez creados se muestran en la pantalla principal para recordarle siempre que el esfuerzo y dedicación tienen recompensa.

Ilustración 46 Progresiones y logros

6.2.4.2 Gestionar entrenamientos

Esta parte es similar a la del profesor solo que el usuario alumno no tendrá que especificar ni el colegio, ni el grupo ni el curso.

6.2.4.3 Gestionar sesiones

En este apartado se permite al usuario grabar entrenamientos y subirlos a la plataforma de vídeo online *YouTube*. Como en apartados anteriores si lo hace desde un ordenador podrá seleccionar el fichero para subirlo, si lo hace desde un dispositivo móvil se activará la cámara.

Una vez subido el vídeo a la plataforma, se mostrará un icono representativo en la página principal que permite ver el vídeo grabado.

Desde esta pantalla se puede notificar al profesor de que hay una sesión disponible para ver.

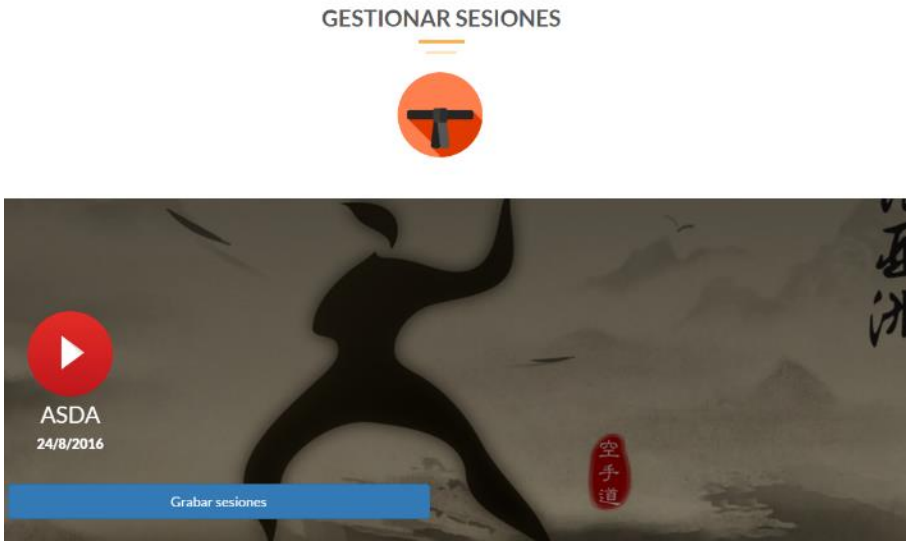


Ilustración 47 Gestionar sesiones

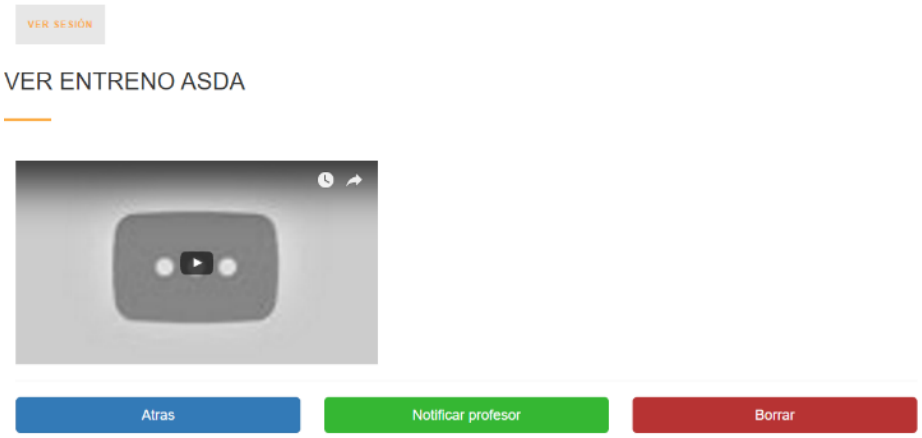


Ilustración 48 Ver sesión

Como se puede ver en la ilustración 48, además de la notificación al profesor también se permite borrar la sesión.

6.2.4.4 Gestionar wazas

En este apartado el usuario alumno podrá crear sus *Wazas* para compartirlas con el profesor y/o subirlas a *Google Drive*.

Pulsando sobre "crear *Wazas*" permitirá elegir qué tipo de *Waza* será. Las diferentes posibilidades son: *tsuki* (técnicas de puño), *geri* (técnicas de pierna, patadas), *uke* (paradas) o *uchi* (golpes indirectos; son aquellos que no se realizan con los nudillos, por ejemplo, un codazo o golpear con el canto de la mano).

Una vez elegida la *Waza* a crear se muestra la siguiente interface.

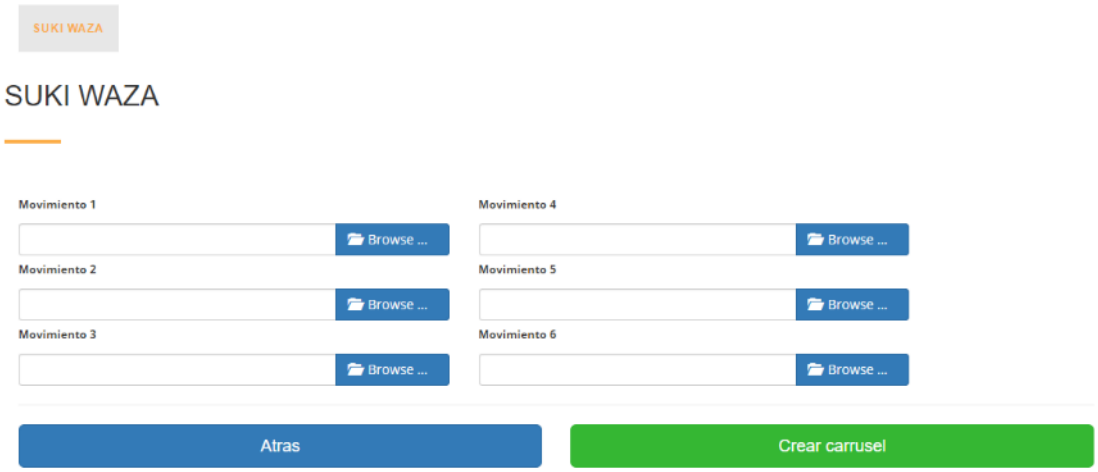


Ilustración 49 Crear waza

En ella, y a través de la cámara, se obtendrán las seis fotos y se mostrará una vista previa de cada una. Si la foto no gusta se puede borrar y sacar otra en su lugar. Una vez se tengan las seis fotos se pulsa "crear carrusel" el cual guarda las fotos en el servidor y se crea la siguiente pantalla donde se muestra el carrusel.

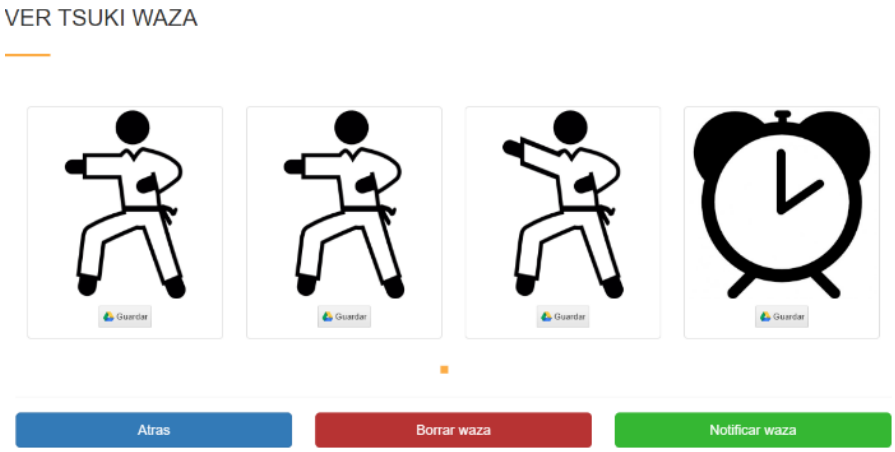


Ilustración 50 Ver waza

En la foto anterior se ve el carrusel, si se desplaza hacia la izquierda muestra las fotos restantes. Se ve debajo de cada foto el botón de *Google Drive* que permite subir la foto a la cuenta personal del usuario de *Drive*.

En esta pantalla se puede borrar el waza por completo o notificar al profesor para que la vea.

6.2.4.5 Foro y perfil

Al igual que el profesor el alumno cuenta con una vista al foro para poder comunicarse con los demás usuarios.

También cuenta con su propia página de perfil para poder cambiar sus datos y contraseña.

6.3 Diseño del código

En este apartado se verá la implementación de las funcionalidades más relevantes, así como de las distintas *APIs* que se incluyen en el proyecto.

6.3.1 Configuración de *CodeIgniter*

El *framework* de *CodeIgniter* permite configurar el sistema de manera muy rápida y sencilla solo modificando el archivo "config" dentro de la carpeta del mismo nombre. Aquí se especificará la URL base para simplificar los direccionamientos web. También la página de inicio por defecto y algunos parámetros más.

```
/*
|--------------------------------------------------------------------------
| Base Site URL
|--------------------------------------------------------------------------
|
| URL to your CodeIgniter root. Typically this will be your base URL,
| WITH a trailing slash:
|
|   http://example.com/
|
| If this is not set then CodeIgniter will try to guess the protocol, domain
| and path to your installation. However, you should always configure this
| explicitly and never rely on auto-guessing, especially in production
| environments.
|
*/
$config['base_url'] = 'http://localhost/karate';
```

Ilustración 51 Ejemplo config

Otro archivo importante es "database" que permite crear la conexión con la base de datos para tenerla siempre disponible. Esto ahorra tener que abrir la conexión por cada función que quiera hacer uso de la base datos.

En este archivo se rellenarán los campos para poder acceder a la base de datos. El usuario, nombre donde está alojada y el Password son algunos de los campos necesarios para la correcta configuración de la base de datos en el *framework.*

```
$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
$db['default']['database'] = 'karate';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;


/* End of file database.php */
/* Location: ./application/config/database.php */
```

Ilustración 52 Ejemplo configuración

En los apartados siguientes veremos cómo se hace uso de los modelos y cómo se abre una conexión a la base de datos que haga uso de esta configuración.

6.3.2 Tabla de alumnos

Volviendo a la pantalla del profesor donde está la tabla con todos los alumnos. La creación de la tabla no es distinta a la de cualquier tabla en HTML, pero lo interesante es la capacidad de filtrar los resultados e incluso combinarlos usando varios filtros a la vez.

En este caso es necesario ampliar el código HTML con JavaScript para poder usar los filtros múltiples.

```
<script language='JavaScript'>
$(document).ready(function() {
    $('#example').DataTable( {
        initComplete: function () {
            this.api().columns().every( function () {
                var column = this;
                var select = $('<select><option value=""></option></select>')
                    .appendTo( $(column.footer()).empty() )
                    .on( 'change', function () {
                        var val = $.fn.dataTable.util.escapeRegex(
                            $(this).val()
                        );

                        column
                            .search( val ? '^'+val+'$' : '', true, false )
                            .draw();
                    } );

                column.data().unique().sort().each( function ( d, j ) {
                    select.append( '<option value="'+d+'">'+d+'</option>' )
                } );
            } );
        }
    } );
} );
</script>
```

Ilustración 53 Tabla filtrable

Este código se encarga de recoger dinámicamente los valores que se agrupan en cada columna para mostrarlos en el selector que se encuentra abajo. Por ejemplo, si en la columna "colegios" tenemos; San Mames, Deusto y Fadura, el selector de abajo contendrá esos valores y si elegimos uno filtrará la tabla para ese resultado.

6.3.3 Dar de alta nuevo alumno

La novedad en este apartado sobre otros registros de usuarios es la capacidad de sacar una foto al momento y ver una vista previa. El botón activará la cámara en caso de estar con el móvil.

El código que permite esto es una sencilla etiqueta de HTML 5.

```
<li>
    <label class="control-label">Select File</label>
    <input id="input-1" type="file" class="file" accept="image/*" capture="camera" name="imagenes" required>
</li>
```

Ilustración 54 Capturar imagen

En esta etiqueta se añade el apartado “accept =” image/*”” para que solo acepte imágenes. Esto hace que en el móvil la cámara se active en modo cámara de fotos y no permita grabar vídeos.

El comando “capture = ”camera”” es el encargado de abrir la cámara del móvil, sin este comando se abriría el explorador de fichero al igual que en el ordenador.

Una vez obtenida la foto, se muestra una previsualización en pantalla para ver si la foto es correcta. Si los campos han sido ya rellenados y se pulsa sobre “aceptar” el sistema guardará el alumno en la base de datos.

Se aprovechará este ejemplo para ver la forma en la que se comunican la vista con el controlador y éste a su vez con el modelo.

Una vez pulsado “aceptar” se envía el formulario al controlador encargado de gestionarlo, en este caso “users”. Dentro del controlador se llama a la función “nuevo_alumno”.

```
function nuevo_alumno(){
    $nombre=$this->input->post('nombre_alum');
    $apellido=$this->input->post('apellido_alum');
    $edad=$this->input->post('edad_alum');
    $colegio=$this->input->post('colegio_alum');
    $curso=$this->input->post('curso_alum');
    $grupo=$this->input->post('grupo_alum');
    $file= $_FILES['imagenes'];
    $id_profe =$this->session->userdata('ID');
    $ruta_imagen='uploads/'.time().$nombre.$apellido;

    move_uploaded_file($file['tmp_name'], $ruta_imagen);

    $this->register_model->insertar_nuevo_alumno($nombre,$apellido,$edad,$colegio,$curso,$grupo,$id_profe,$ruta_imagen);

    $this->session->set_flashdata('mensaje', 'Usuario registrado correctamente!');
            redirect('../index.php?/users/new_alumno', 'refresh');


}
```

Ilustración 55 Controlador users

Como se ve el controlador recibe los parámetros a través del método “post” y una vez los recibe, carga la foto. Para el nombre de la foto y a fin de evitar problemas con dos usuarios que se llamen igual, se le añade la función “time()” que inserta la hora del sistema.

El comando “move_uploaded_file” sube la imagen a la carpeta especificada en el servidor; en este caso “Uploads”

A continuación se muestra una parte interesante, que es cómo el controlador se comunica con el modelo.

```
$this->register_model->insertar_nuevo_alumno($nombre,$apellido,$edad,$colegio,$curso,$grupo,$id_profe,$ruta_imagen);
```

Ilustración 56 Conexión modelo

Usando “register_model” se llama al modelo con el mismo nombre, y busca la función “insertar_nuevo_alumno” al cual le pasa como parámetros todos los datos del alumno.

Ya en el modelo se encuentra lo siguiente:

```
function insertar_nuevo_usuario($nombre,$apellido,$email,$cinturon,$ano,$dojo,$password,$rol,$cod_profesor){
    $sql = "INSERT INTO `kt_users`(`Nombre`, `Apellidos`, `Email`, `Cinturon`, `Ano_com`, `Rol`, `Dojo`,`Cod_profesor`, `Password`) VALUES ('".$nombre."','".$apellido."','".$email."','$cinturon','$ano'
    $this->db->query($sql);

}
```

Ilustración 57 Modelo registro

Se define la función *SQL (*en este caso será un *INSERT) y* en la segunda línea se ejecuta. Una vez ejecutada la función el control vuelve al controlador, que redireccionará a otra pantalla para mostrar el mensaje de confirmación.

Como se puede ver en referencia a lo mencionado en el punto anterior, gracias a *CodeIgniter* y su archivo de configuración de la base de datos se tiene que abrir y cerrar la conexión cada vez que se quiera hacer una consulta.

6.3.4 Asistencia

Esta es una de las funcionalidades más importantes en la web. Según el desplazamiento que se ejecute sobre la foto se marcará al alumno como presente o ausente.

Esto se consigue con JavaScript. En la próxima captura se muestra el código.

```
<script>
$(document).on("pagecreate","#pageone",function(){
  $("img").on("swipeleft",function(){
    $("span").text("Alumno en clase");
    var pos=<?php echo $pos?>;
    var colegio ="<?php echo($alumnos[0]->Colegio)?>";
    var curso =<?php echo($alumnos[0]->Curso)?>;
    var grupo =<?php echo($alumnos[0]->Grupo)?>;
    var id=<?php echo($alumnos[$pos]->ID)?>;
    console.log(id);
    window.location="index.php?/asistencia/esta/"+colegio+"-"+curso+"-"+grupo+"-"+pos+"-"+id;
  });
});
$(document).on("pagecreate","#pageone",function(){
  $("img").on("swiperight",function(){
    $("span").text("Alumno ausente");
    var pos=<?php echo $pos?>;
    var colegio ="<?php echo($alumnos[0]->Colegio)?>";
    var curso =<?php echo($alumnos[0]->Curso)?>;
    var grupo =<?php echo($alumnos[0]->Grupo)?>;
    var id=<?php echo($alumnos[$pos]->ID)?>;
    console.log(id);
    window.location="index.php?/asistencia/no_esta/"+colegio+"-"+curso+"-"+grupo+"-"+pos+"-"+id;
  });
});
</script>
```

Ilustración 58 Swipe

Se puede ver que, si el profesor desplaza a un lado u otro se ejecuta la función correspondiente, “swiperight” o “swipeleft”. Ambas ejecutan un código similar, pero al final envía los datos a una función distinta que es la encargada de registrar los datos en la base de datos. Una función guarda si el alumno está presente y la otra si está ausente ese día.

6.3.5 Examen

Otra funcionalidad requerida es la posibilidad de realizar exámenes. Después de ver la interface en el apartado anterior, se verá el código detrás de los círculos de puntuación (los círculos azules que se muestran en la interface y permite puntuar una de las cuatro áreas de examen).

Como en el apartado anterior, estos círculos se gestionan mediante JavaScript.

```
<script type="text/javascript">
    $("#slider1").roundSlider({
        sliderType: "min-range",
        handleShape: "round",
        width: 22,
        radius: 100,
        value: 50,
        editableTooltip: false,
        change: function (e) {

                t=e.value;
            },
        beforeCreate: function(e){
            t=50;
        }
    });
```

Ilustración 59 Círculo de puntuación

Para el funcionamiento de este JavaScript es necesario tener otros dos archivos JavaScript guardados y enlazados a la página donde se encuentre este código. En la captura siguiente vemos cómo enlazar a un archivo JavaScript desde la vista PHP.

```
<link href="css/roundslider.min.css" rel="stylesheet" >
<script src="js/roundslider.min.js"></script>
```

Ilustración 60 Conexión JavaScript

En esta última captura vemos que además del archivo JavaScript que da las funcionalidades necesarias para su correcto funcionamiento, también se incluye el archivo ".CSS" que le da el formato visual al círculo.

Volviendo a la ilustración 59, se muestra cómo la función llama a "roundSlider" que es una función interna del JavaScript "roundSlider". Los siguientes parámetros son la configuración visual del círculo, como la anchura del radio y el valor inicial que mostrará.

La función "change" ha sido creada para la ocasión y se ejecuta cada vez que el círculo es modificado a otro valor. Por ejemplo, si se cambia de 50 a 70 se ejecuta esa acción, guardando en la variable "t" el valor actual del círculo o sea 70.

El círculo presentaba un problema; si no se modificaba y se intentaba capturar su valor, este daba 0. Esto significa que por defecto el círculo muestra el valor 50, y supongamos que el alumno saca un 50/100 en esa parte, no es necesario modificar el círculo. A la hora de capturar el valor con la función "change", evidentemente no funcionaba. Esto es debido al punto anterior, el valor solo se modificaba cuando el círculo cambiaba su valor. Si no se tocaba, la variable no se definía y no se mostraba nada.

Para corregir ese problema se creó la función "beforeCreate". Esta función se ejecuta nada más crear el círculo asignándole un valor a la variable "t" de 50. Si el círculo no se modifica entonces la variable permanece a 50 y graba ese valor cuando se le requiera por el controlador. Si el círculo es modificado, se llama a la función "change" y guarda el valor nuevo.

6.3.6 *API* de *YouTube*

A partir de aquí ya se entra en las *APIs*. En este primer apartado se va a explicar la implementación de la *API* de *YouTube.*

En primer lugar, hay que registrar la *API* en la página de *Google Developers* y crear un nuevo proyecto. En ese nuevo proyecto se ha de elegir la aplicación que se va a usar, en este caso *YouTube*. Una vez elegida se han de configurar varios parámetros para poder obtener el "id" que necesitará la aplicación para funcionar.

Se han de especificar también desde qué página se llamará al *API* y cual recibirá los datos de vuelta. Se enseña una captura a continuación. No obstante, no se puede ver nada dado que todos los parámetros han de ser secretos.

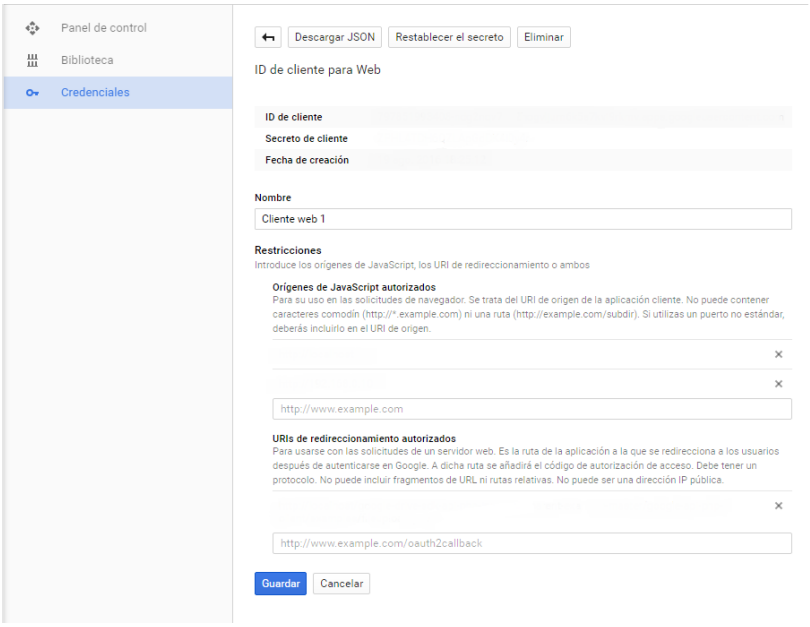


Ilustración 61 Credenciales YouTube

Esta *API* se encuentra en dos apartados del proyecto, en la funcionalidad de realizar exámenes por parte del profesor y en grabar sesiones por parte del usuario alumno.

No obstante, comparten funcionamiento así que se explicará solamente uno de ellos.

Volviendo a la pantalla de examen que se ha mostrado anteriormente (ilustración 41), se observa debajo de los campos nombre y apellido del alumno el nombre "jonathan guijarro". Esto es debido a que el sistema está identificado con este usuario.

Si el usuario tiene la sesión iniciada de *Google* en su navegador, el sistema le preguntará la primera vez si quiere dar permisos de subida a la página. Si no está identificado, se abrirá una ventana donde se pedirá que inserte su usuario y contraseña de *Google.* Una vez insertados se mostrará el nombre del propietario de la cuenta.

Todo esto se realiza a través de *Google* y su pasarela, sin tener que guardar en el proyecto ningún dato de usuario ni de contraseña de *Google* o *YouTube*.

Se puede cambiar la privacidad del vídeo a subir entre pública o privada. Esto también está gestionado por *Google.*

A continuación, vemos el botón de seleccionar archivo; el cual funciona similar a los descritos arriba para otras funcionalidades. La única diferencia es que esta vez abrirá la cámara en modo vídeo dispuesta a grabar.

Esto se consigue modificando la etiqueta de HTML 5 como se muestra a continuación.

```
<input input type="file" id="file" capture="camera"class="button" accept="video/*">
```

Ilustración 62 Capturar vídeo

En este caso se establece "accept="vídeo/*"" para especificar que se quiere un fichero de formato vídeo o la cámara en modo vídeo en el caso de estar en un dispositivo móvil.

Una vez se tiene el vídeo precargado, se podrá subir a *YouTube* mediante el botón "upload vídeo". Se mostrará una barra de progreso según el vídeo se vaya subiendo. Una vez subido, se dará el enlace para poder verlo.

La mayor parte del código está proporcionado por *Google* y deja muy poco margen a la modificación. Se modificó el input para acceder a la cámara en modo vídeo y algún apartado más. A continuación, se muestra el código completo.

```
<span id="signinButton" class="pre-sign-in">
  <!-- IMPORTANT: Replace the value of the <code>data-clientid</code>
       attribute in the following tag with your project's client ID. -->
  <span
    class="g-signin"
    data-callback="signinCallback"
    data-clientid="
    data-cookiepolicy="single_host_origin"
    data-scope="https://www.googleapis.com/auth/youtube.upload https://www.googleapis.com/auth/youtube">
  </span>
</span>

<div class="post-sign-in">
  <div>
    <span id="channel-name"></span>
    <br>
  </div>

  <div>
    <input id="title" style="display:none;" type="text" value="<?=$alumno->Nombre.' '.$alumno->Apellidos?>">
  </div>
  <div>
    <textarea  style="display:none;" id="description"><?=$alumno->Nombre.' '.$alumno->Apellidos?> Examen</textarea>
  </div>
  <div>
    <label for="privacy-status">Privacy Status:</label>
    <select id="privacy-status">
      <option>public</option>
      <option>unlisted</option>
      <option>private</option>
    </select>
  </div>

  <div>
    <input input type="file" id="file" capture="camera"class="button" accept="video/*">
    <button class="btn btn-lg btn-secondary btn-block" id="button">Upload Video</button>
  <div class="during-upload">
    <p><span id="percent-transferred"></span>% done (<span id="bytes-transferred"></span>/<span id="total-bytes"></span> bytes)</p>
    <progress id="upload-progress" max="1" value="0"></progress>
  </div>

  <div class="post-upload">
    <p>Uploaded video with id <span id="video-id"></span>. Polling for status...</p>
    <ul id="post-upload-status"></ul>
    <div id="player"></div>
  </div>
</div>
```

Ilustración 63 Api YouTube

En la captura anterior se ha borrado el "data-clientid" puesto que es un apartado personal.

Como se puede apreciar en el código, se comienza con un "span" que contiene la clase que hace funcionar todo: "class =pre-sing-in". Se modificaron algunos campos para evitar la interacción del usuario, por ejemplo, en vez de dejar que el usuario escogiera el título para el vídeo, se guarda automáticamente un título prefijado.

El botón que sube el vídeo y la barra la controla totalmente la *API.*

Como se ha mencionado en el "script" anterior, esta *API* necesita de archivos ". js" para su funcionamiento. Estando enlazados de esta manera en la página.

```
<script src="jsy/cors_upload.js"></script>
<script src="jsy/upload_video.js"></script>
```

Ilustración 64 Enlace API YouTube

Como se mencionó en otro apartado, están guardados en una carpeta separada del resto de ".js". Una carpeta llamada "jsy".

6.3.7 *API de Google Drive*

Como se ha visto en apartados anteriores, en un primer momento se eligió *Dropbox* como servicio de alojamiento de archivos, pero la imposibilidad de adaptarlo al sistema exigió la necesidad de explorar otras opciones.

Por ello se eligió *Google Drive* como nueva opción, dado que se tenía cierta experiencia con la *API* de *YouTube*. No obstante, también se presentaron multitud de dificultades para adaptarlo en el proyecto.

Al final se encontró una opción, si bien no es la mejor para el proyecto, es la que mejor se adapta a las funcionalidades de la web. Esta solución pasa por colocar un botón debajo de cada foto y el usuario puede elegir qué foto subir. Una vez pulsa el botón se le abrirá una página de *Google* a su *Drive* y podrá elegir dónde guardar la foto. No es la mejor solución puesto que se hubiera preferido subir todas las fotos juntas en vez de una en una, pero esta solución también permite quedarse solo con las mejores sin guardar el resto.

El botón se crea mediante este código

```
<div class="caption">
    <div class="g-savetodrive"
        data-src="<?=$imagen4?>"
        data-filename="Movimiento 4"
        data-sitename="My">
    </div>
</div>
```

Ilustración 65 Código Google Drive

Al igual que apartados anteriores, necesita de un ".js" para funcionar solo que esta vez no lo descarga de un servidor local sino de una copia remota en *Google.com*, lo cual permite mantener "cacheados" los scripts (evitando su múltiple descarga en casos en los que distintas webs reutilicen el mismo script remoto). Tiene la siguiente etiqueta HTML:

```
<script src="https://apis.google.com/js/platform.js" async defer></script>
```

Ilustración 66 API Google Drive

6.3.8 *API* de *VanillaForum*

Esta *API* está presente para el usuario profesor y el usuario alumno, permite la comunicación entre ellos y los demás usuarios de la página. Se pueden crear temas para tratar entre todos o usar mensajes privados para hablar solo con una persona.

La *API* requiere de una instalación previa; no basta con bajar los archivos y enlazarlos como el resto (esto es debido a que necesita crear una base de datos propia). Después de configurar todo, *Vanilla* crea multitud de tablas en la base de datos para el funcionamiento interno del foro, tablas que para una configuración sencilla no es necesario tocar.

Para mostrar el foro en una pantalla tenemos que activar el modo embebido y nos facilita este "script".

```
<script type="text/javascript" src="<?php echo base_url() . 'vanilla/js/embed.js';?>"></script>
```

Ilustración 67 Script VanillaForum

El "script "llama a una función dentro de un ".js" en la instalación de *Vanilla.*

Es necesario instalar dos "plugins" para automatizar el proceso de logeo en el foro. Sin este añadido el foro funciona, pero no entra automáticamente a la sesión de la persona que está identificada en la web.

Un "plugin" es el "jsconnect"; el cual modificando su configuración permite redireccionar la entrada al foro usando las credenciales de la web que lo aloja.

También es necesario el "jsauto-connect"; el cual añade una mejora al anterior y hace que esa entrada al foro sea automática sin tener que pulsar ningún botón. En cuanto el usuario entre en el foro, se detectarán sus credenciales automáticamente (sin necesidad de volverlas a introducir).

6.4 Responsive

Como se ha mencionado a lo largo de todo el documento, la web es *Responsive* lo cual significa que tiene que adaptarse a diferentes tamaños de pantalla y dispositivos.

A continuación, se muestran algunas capturas desde una *tablet* de siete pulgadas para ver la diferencia con la versión de ordenador de escritorio.

Ilustración 68 Responsive ej.1

Ilustración 69 Responsive ej.2

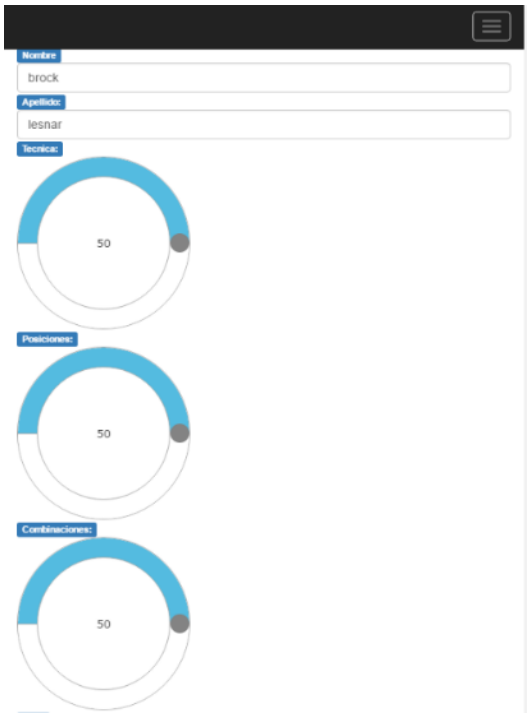


Ilustración 70 Responsive ej.3

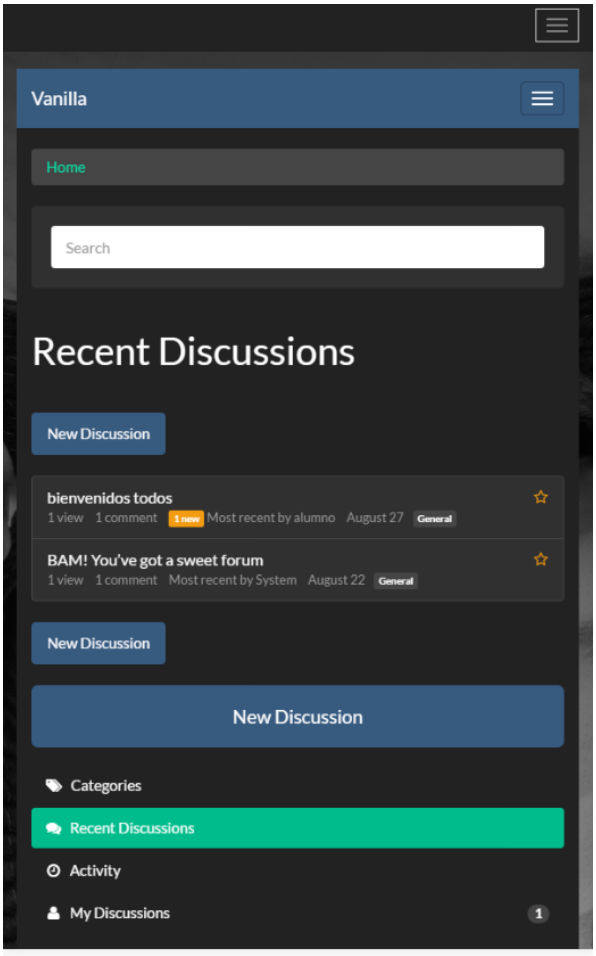


Ilustración 71 Responsive ej.4

7 Verificación y pruebas

En este apartado se documentarán las pruebas realizadas durante la creación de la web a fin de corregir todos los posibles problemas que puedan aparecer y obtener un producto de mejor calidad.

7.1 Objetivo

El objetivo de las pruebas es verificar el correcto funcionamiento de cada prototipo de manera individual y en conjunto con el resto de funcionalidades. Si una prueba reporta un comportamiento no deseado, se procede a la reparación del código para que se ajuste a los requisitos establecidos con anterioridad.

7.2 Registrar usuario

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
1.	Registrar un usuario profesor nuevo.	Mensaje de confirmación.	Mensaje.	Correcto.
2.	Registrar un usuario profesor que ya existe en la base de datos.	Mensaje indicando que el usuario ya existe.	Mensaje.	Correcto.
3.	Intentar dejar campos en blanco (excepto código de profesor) durante el registro.	Notificación indicando que el campo no se puede dejar en blanco	Notificación.	Correcto.
4.	Introducir una contraseña distinta en los campos de verificación de la contraseña.	Mensaje de advertencia indicando que las contraseñas son distintas.	Mensaje.	Correcto.
5.	Registrar un usuario alumno incluyendo el código de profesor.	Mensaje de confirmación.	Mensaje.	Correcto.

Tabla 3 Registrar usuario

7.3 Login

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
6.	Logear un usuario con la email o el Password incorrecto.	Mensaje indicando que los datos son incorrectos.	Mensaje.	Correcto.
7.	Logear un usuario con email y Password correctos	Acceso a la pantalla principal de la web.	Acceso a la pantalla principal.	Correcto.

Tabla 4 Login

7.4 Nuevo alumno

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
8.	El profesor da de alta un nuevo alumno.	Alumno dado de alta en el sistema y un mensaje de confirmación.	Alumno dado de alta y mensaje de confirmación.	Correcto.
9.	El alumno nuevo figura en la tabla de alumnos de la página principal.	La tabla se actualiza con el nuevo alumno.	Tabla actualizada	Correcto.
10.	El profesor da de alta un nuevo alumno dejando campos sin rellenar.	Notificación indicando que el campo no se puede quedar en blanco.	Notificación.	Correcto.
11.	El sistema inicia la cámara en un dispositivo móvil para obtener una foto del alumno.	Inicio de la aplicación cámara.	Aplicación cámara iniciada.	Correcto.

Tabla 5 Nuevo alumno

7.5 Ver alumno

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
12.	Muestra la información del alumno.	Ver por pantalla la información del alumno.	Pantalla con información del alumno.	Correcto.

Tabla 6 Ver alumno

7.6 Editar alumno

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
13.	El profesor cambia alguno de los datos del alumno.	Se actualiza la información del alumno y se muestra mensaje de confirmación.	Alumno actualizado y mensaje de confirmación.	Correcto.
14.	El profesor borra al alumno.	Se borra del alumno del sistema y se muestra un mensaje de confirmación.	Alumno borrado y mensaje de confirmación.	Incorrecto.

Tabla 7 Editar alumno

En el apartado 14 se pide borrar al alumno y mostrar un mensaje de confirmación, en este caso el alumno se borra correctamente pero el mensaje presenta errores que no ayudan al usuario a saber que ha ocurrido. Para reparar el error se creó un nuevo mensaje que se muestra en la pantalla principal con una descripción más acorde.

7.7 Gestión de alumnos

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
15.	El profesor selecciona el colegio en el desplegable.	Se muestran todos los cursos y grupos del colegio elegido.	Pantalla con los cursos y grupos del colegio elegido.	Correcto.

Tabla 8 Gestión alumnos

7.8 Control de asistencia

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
16.	Se muestra la foto del alumno y se desliza hacia la izquierda. (alumno ausente)	Guardar en el la base de datos que el alumno está ausente. Y mostrarlo en la tabla de asistencias.	Alumno guardado en la base de datos como ausente y reflejado en la tabla de asistencia.	Correcto.
17.	Se muestra la foto del alumno y se desliza hacia la derecha. (alumno presente)	Guardar en el la base de datos que el alumno está presente. Y mostrarlo en la tabla de asistencias.	Alumno guardado en la base de datos como presente y reflejado en la tabla de asistencia.	Correcto.

Tabla 9 Control asistencia

7.9 Tabla de asistencia

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
18.	Una vez pasada la asistencia a un alumno se deberá reflejar en la tabla de asistencia.	Mostrar en la tabla de asistencia si el alumno fue o no a clase.	Tabla con las asistencias de los alumnos.	Correcto.
19.	La tabla deberá mostrar solo los días del mes actual	Mostrar la tabla de asistencias del mes en curso.	Tabla de asistencias totales.	Incorrecto.

Tabla 10 Tabla asistencia

A la hora de mostrar la asistencia de cada alumno, se muestran los días de todos los meses, dando lugar a errores. Esto no es lo que se busca, interesa que solo muestre los datos del mes en curso para una mejor legibilidad.

Para subsanar este error en el modelo que filtra los datos de cada alumno se inserta una cláusula para que filtre también por mes y año.

7.10 Tabla examen

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
20.	Muestra todos los alumnos de la clase. Si han realizado el examen muestra la nota y la fecha en la que se hizo.	Tabla con alumnos y sus datos del examen.	Tabla con datos del alumno.	Correcto.

Tabla 11 Tabla examen

7.11 Examen

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
21.	El profesor realiza el examen al alumno sin modificar los círculos de puntuación.	Se guarda el examen del alumno con todos sus datos.	Se guarda el examen pero los círculos no se guardan correctamente.	Incorrecto.
22.	El profesor realiza el examen del alumno modificando los círculos de puntuación.	Se guardan el examen del alumno con todos los datos.	Todos los datos se guardan correctamente.	Correcto.
23.	Una vez realizado el examen carga automáticamente el examen del siguiente alumno.	Aparición del examen del siguiente alumno.	Se muestra por pantalla el examen del siguiente alumno.	Correcto.

Tabla 12 Examen

Como se mencionó en el apartado referente al código de esta sección, este fallo se debía a que el valor por defecto de los círculos (50) no se cogía y necesitaba ser cambiado a otro valor para funcionar. Se arregló añadiendo una nueva función “beforeCreate” que capturaba el valor del círculo seguido a su creación.

7.12 Ver examen

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
24.	Se mostrarán por pantalla los datos del examen realizado por el alumno.	Datos del examen del alumno.	Datos del alumno pero con error en el formato.	Incorrecto.

Tabla 13 Ver examen

Una vez realizado el examen se puede visualizar los datos guardados. En uno de los apartados los datos se mostraban de una manera errónea, mostrando en lugar de un espacio la siguiente cadena "%20". Esta cadena representa un espacio en PHP. Para corregir ese fallo, se usa la función "str_replace" que permite cambiar una cadena de texto por otra.

7.12 Añadir entrenamientos

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
25.	El profesor crea un nuevo entrenamiento para un grupo en concreto.	Se da de alta la información y se muestra un icono en la página principal del profesor.	Datos creados e icono en página principal del profesor.	Correcto.
26.	El alumno crea un nuevo entrenamiento.	Se da de alta la información y se muestra un icono en la página principal del alumno	Datos creados e icono en página principal del alumno.	Correcto.

Tabla 14 Añadir entrenamientos

7.13 Tabla de entrenamientos

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
27.	El profesor elige un grupo y la web le muestra todos los entrenamientos asignados a ese grupo.	Una tabla con todos los entrenamientos de ese grupo.	Tabla con los entrenamientos.	Correcto.

Tabla 15 Tabla de entrenamientos

7.14 Ver entrenamientos.

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
28.	El profesor puede ver y editar un entrenamiento ya creado anteriormente.	Actualizar los datos del entrenamiento.	Datos del entrenamiento actualizados.	Correcto.
29.	El profesor puede borrar el entrenamiento.	Borrar en la base de datos y desaparece el icono de la página principal.	Datos borrados e icono quitado.	Correcto.

Tabla 16 Ver entrenamientos

7.15 Foro

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
30.	Mostrar el foro en la página principal.	Mostrar el foro.	Se muestra el foro en la página principal.	Correcto.
31.	Se comunican el profesor y el alumno a través del foro.	Comunicación entre ambos roles.	Hay comunicación entre ambos roles.	Correcto.
32.	El foro muestra siempre la página principal donde se ven las conversaciones.	Página principal del foro.	En ocasiones muestra una página que no es la principal.	Incorrecto.

Tabla 17 Foro

El foro está configurado para mostrar siempre la página principal. En ella se ven todas las conversaciones que hay actualmente creadas. El problema viene cuando alguna página redirecciona a una sección de la web principal. Estas secciones incluyen el carácter "#" y esto causa que el foro no entienda la dirección y muestre una página que no es la principal.

Para solucionarlo se redirecciona siempre al comienzo de la página principal de la web.

7.16 Perfil profesor

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
33.	Se mostrarán los datos personales del profesor que podrá modificar si así lo desea.	Actualización de los datos del profesor y mensaje de confirmación.	Datos del profesor actualizados y se muestra el mensaje de confirmación.	Correcto.
34.	El profesor introduce el colegio, el curso y el grupo correctamente para borrar los exámenes.	Se borran todos los exámenes de ese grupo y se muestra un mensaje.	Se borran los exámenes y se muestra el mensaje.	Correcto.
35.	El profesor introduce el colegio, el curso y el grupo erróneamente para borrar los exámenes.	Se muestra un mensaje de error.	Se muestra el mensaje.	Correcto.

Tabla 18 Perfil profesor

7.17 Logros y progresión

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
36.	El alumno podrá apuntar un logro o la obtención de un cinturón que será mostrado en la vista principal.	Se guardan los datos y se crea el icono en la vista principal.	Se guardan los datos y se crea el icono.	Correcto.

Tabla 19 Logros y progresión

7.18 Grabar sesión alumno

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
37.	El alumno podrá grabar una sesión y subirla a *YouTube*.	Se mostrará un icono en la pantalla principal indicando que hay una sesión grabada.	Se guardan los datos y aparece el icono.	Correcto.

Tabla 20 Grabar sesión alumno

7.19 Ver sesión

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
38.	El alumno podrá ver la sesión que ha grabado	Se mostrará una pantalla con el vídeo.	Se obtiene una pantalla con el vídeo de la sesión.	Correcto.

Tabla 21 Ver sesión

7.20 Notificar sesión

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
39.	El alumno podrá notificar al profesor que ha grabado una sesión para que este la vea.	Se mostrará el apartado notificaciones al profesor con el icono de la sesión.	Se obtiene una pantalla con el vídeo de la sesión.	Correcto.

Tabla 22 Notificar sesión

7.21 Revisar notificación

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
40.	El profesor recibe la notificación de la sesión o de la *waza*. Y la puede marcar como vista para quitar el aviso.	Se acepta la notificación y desaparece el icono de aviso.	El icono desaparece una marcada como vista la notificación.	Correcto.

Tabla 23 Revisar notificación

7.22 Diseño

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
41.	El diseño se adapta a todos los tamaños de pantalla.	El diseño se auto-dimensionara para mostrarse correctamente en todos los tamaños de pantalla.	En algunas pantallas el tamaño no era el correcto.	Incorrecto.
42,	El menú se recoge correctamente según se baja por la página.	Menú recogido.	Menú se queda desplegado.	Incorrecto.

Tabla 24 Diseño

En algunas pantallas el *Responsive* no se adaptaba perfectamente al tamaño de la pantalla. Este problema se reparó mirando clases de PHP que fueran no *Responsive* y sustituyéndolas por otras que si lo fueran.

El menú superior se comprime según se desciende por la página, en la *tablet* daba problemas y se quedaba desplegado. Se corrigió suprimiendo un archivo ".js" que interfería con el que se encargaba de controlar el menú superior.

7.23 Seguridad

Código	Descripción	Resultado esperado	Resultado obtenido	Comentarios
43.	El sistema expulsa al usuario si pasa demasiado tiempo sin navegar.	Enviar al usuario a la pantalla de login	No expulsa al usuario	Incorrecto
44.	El sistema evita que un usuario sin estar identificado acceda a ciertas páginas mediante la URL	Enviar al usuario a la pantalla de login	Muestra el contenido de la pantalla aun no permitiéndole interactuar	Incorrecto.

Tabla 25 Seguridad

Ambos errores se corrigieron aplicando una función de seguridad en todos los controladores que comprueba si el usuario este identificado en ese momento.

8 LOPD

Debido a que esta web almacena fotos y vídeos de los usuarios/alumnos, también muy posiblemente menores de edad, es necesario hacer un pequeño hincapié en este apartado.

Los datos que se van a almacenar en esta web son datos personales y por tanto se ha de cumplir con la Ley Orgánica de Protección de Datos(LOPD) debido a que incluso el email ya se considera dato personal.

Para evitar cualquier contratiempo legal la página web deberá mostrar un mensaje de confirmación a la hora de registrarse informando qué datos se van a guardar. El usuario deberá aceptar estas condiciones para poder proseguir con el registro.

También se deberá tener un apartado donde el usuario pueda ver un aviso legal sobre el tratamiento de los datos.

Referente a las fotos de los alumnos menores se necesitará un consentimiento externo que deberán firmar los padres para que el profesor pueda sacar una foto al alumno. De no tener el consentimiento se podrá seguir usando la aplicación, pero con una foto genérica.

9 Conclusiones

En este último apartado se va a comparar la planificación inicial y la que finalmente ha sido, el futuro para KuroObi y una pequeña reflexión final.

9.1 Planificación inicial Vs planificación real.

La mayor diferencia ha sido en la planificación temporal, cosa que era muy predecible ya que como se menciono era la primera vez que se realiza un trabajo de esta envergadura.

Se estimó una duración total de 12 semanas trabajando una media de 4 horas diarias durante siete días. La duración sí ha sido correctamente estimada pero no las horas trabajadas por día, siendo muy superior a 4 horas. Por otro lado, no se ha trabajado todos los días incluso ha habido una semana de vacaciones donde no se trabajó nada.

En este apartado también difieren las pruebas de cada prototipo puesto que no se realizaron una vez se tenían todos los prototipos, sino que se realizaban según se terminaba el prototipo en el que se estaba trabajando. De esta manera, según se terminaba se testeaba para obtener y corregir los fallos que pudieran salir y dar el prototipo como terminado. Al final sí se realizaron pruebas totales para confirmar que todos los prototipos funcionasen correctamente entre ellos.

A continuación, se mostrará una comparativa de la planificación temporal.

Estimacion inicial		*Estimacion real*		*Diferencia*
Tareas	*Estimación de horas*	*Tareas*	*Estimación de horas*	
1-Gestion	**23**	**1-Gestion**	**23**	0
1.1 Captura de requisitos	3	1.1 Captura de requisitos	2	
1.2 Definir proyecto	2	1.2 Definir proyecto	2	
1.3 Definir objetivos	1	1.3 Definir objetivos	1	
1.4 Definir tareas	3	1.4 Definir tareas	2	
1.5 Planificar tareas	3	1.5 Planificar tareas	2	
1.6 Estimar tiempos de tareas	2	1.6 Estimar tiempos de tareas	1	
1.7 Documentación	9	1.7 Documentación	13	
2-Análisis	**33**	**2-Análisis**	**21**	-12
2.1 Definir funcionalidades	20	2.1 Definir funcionalidades	15	
2.2 Seleccionar herramientas	3	2.2 Seleccionar herramientas	3	
2.3 Aprendizaje de las herramientas	10	2.3 Aprendizaje de las herramientas	3	
3- Diseño	**51**	**3- Diseño**	**46**	-5
3.1 Diseño de las interfaces	15	3.1 Diseño de las interfaces	24	
3.2 Diseño de la estructura de la web	30	3.2 Diseño de la estructura de la web	20	
3.3 Diseño de la base de datos	6	3.3 Diseño de la base de datos	2	
4-Implementacion	**260**	**4-Implementacion**	**195**	-65
4.1 Definir prototipos	10	4.1 Definir prototipos	10	
4.2 Implementar prototipo 1	30	4.2 Implementar prototipo 1	20	
4.3 Implementar prototipo 2	45	4.3 Implementar prototipo 2	30	
4.4 Implementar prototipo 3	30	4.4 Implementar prototipo 3	15	
4.5 Implementar prototipo 4	25	4.5 Implementar prototipo 4	30	
4.6 Implementar prototipo 5	20	4.6 Implementar prototipo 5	15	
4.7 Implementar prototipo 6	10	4.7 Implementar prototipo 6	5	
4.8 Implementar prototipo 7	20	4.8 Implementar prototipo 7	10	
4.9 Implementar prototipo 8	40	4.9 Implementar prototipo 8	40	
4.10 Implementar prototipo 9	10	4.10 Implementar prototipo 9	15	
4.11 Implementar prototipo 10	20	4.11 Implementar prototipo 10	5	
5-Plan de pruebas	**65**	**5-Plan de pruebas**	**95**	30
5.1 Elaborar plan de pruebas	15	5.1 Elaborar plan de pruebas	15	
5.2 Corregir errores prototipo 1	5	5.2 Corregir errores prototipo 1	5	
5.3 Corregir errores prototipo 2	5	5.3 Corregir errores prototipo 2	5	
5.4 Corregir errores prototipo 3	5	5.4 Corregir errores prototipo 3	5	
5.5 Corregir errores prototipo 4	5	5.5 Corregir errores prototipo 4	15	
5.6 Corregir errores prototipo 5	5	5.6 Corregir errores prototipo 5	5	
5.7 Corregir errores prototipo 6	5	5.7 Corregir errores prototipo 6	5	
5.8 Corregir errores prototipo 7	5	5.8 Corregir errores prototipo 7	5	
5.9 Corregir errores prototipo 8	5	5.9 Corregir errores prototipo 8	15	
5.10 Corregir errores prototipo 9	5	5.10 Corregir errores prototipo 9	5	
5.11 Corregir errores prototipo 10	5	5.11 Corregir errores prototipo 10	5	
		5.12 Prueba total	10	
6-Documentacion	**70**	**6-Documentacion**	**95**	25
6.1 Escribir memoria	35	6.1 Escribir memoria	60	
6.2 Preparar la defensa	35	6.2 Preparar la defensa	35	
TOTAL	**502**	**TOTAL**	**475**	-27

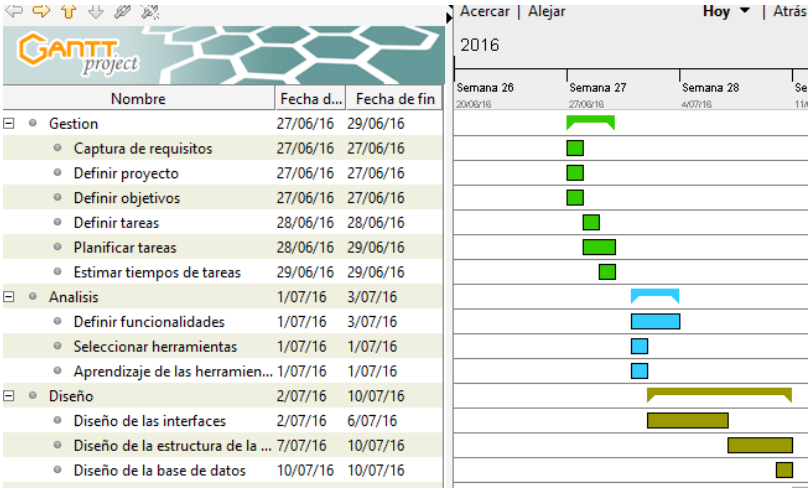
Ilustración 72 Planificación inicial Vs final

Como se puede ver hay una diferencia final de 27 horas menos en cuanto a la estimación inicial. Esto se explica ya que se puso una planificación inicial muy optimista para poder cubrir posibles retrasos. Según se fueron creando los diferentes apartados se fueron ajustando los tiempos dando una duración total de 475 horas.

El cambio más importante ocurre en el apartado de "Implementación" que ha visto reducido su tiempo en 65horas. Se implementó más rápido de lo previsto debido a los conocimientos adquiridos durante las prácticas de empresa donde se ha trabajado PHP, MySQL, JavaScript y HTML5.

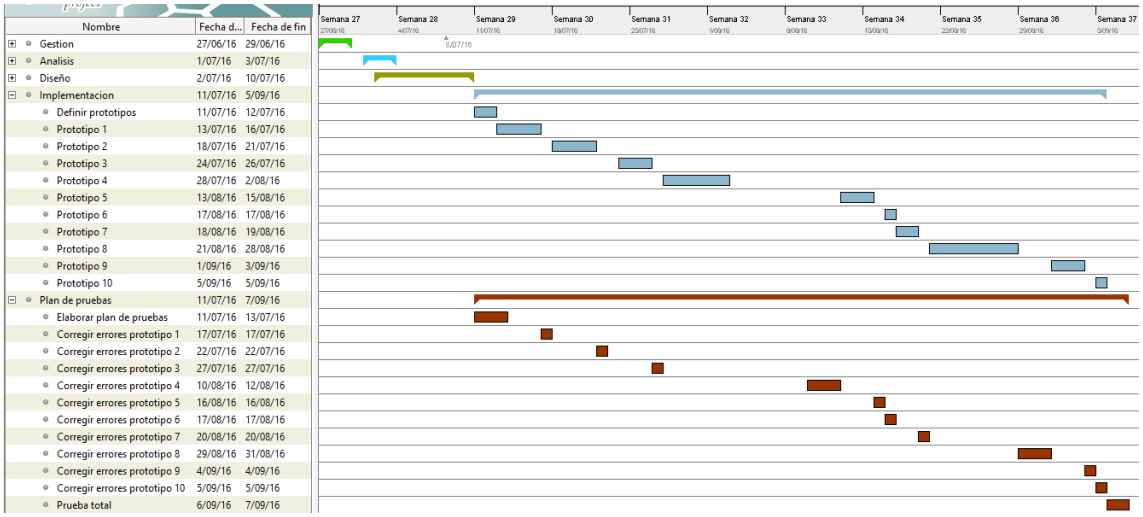
Esto ha dado mucho conocimiento sobre herramientas, forma de programación, conexiones con base de datos y mantenimiento lo cual ha acelerado mucho la implementación del TFG.

A continuación, se muestra un Gantt actualizado a la nueva tabla en contraposición al mostrado en los primeros apartados (ilustración 2 y 3).



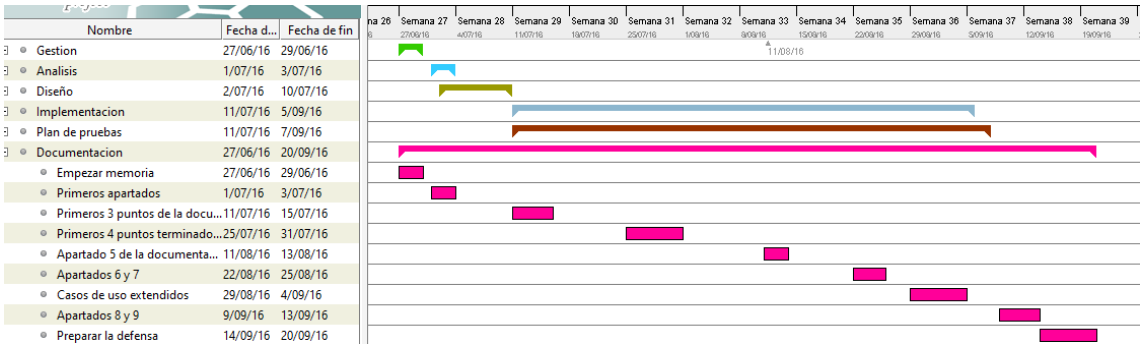
Nombre	Fecha d...	Fecha de fin
Gestion	27/06/16	29/06/16
Captura de requisitos	27/06/16	27/06/16
Definir proyecto	27/06/16	27/06/16
Definir objetivos	27/06/16	27/06/16
Definir tareas	28/06/16	28/06/16
Planificar tareas	28/06/16	29/06/16
Estimar tiempos de tareas	29/06/16	29/06/16
Analisis	1/07/16	3/07/16
Definir funcionalidades	1/07/16	3/07/16
Seleccionar herramientas	1/07/16	1/07/16
Aprendizaje de las herramien...	1/07/16	1/07/16
Diseño	2/07/16	10/07/16
Diseño de las interfaces	2/07/16	6/07/16
Diseño de la estructura de la ...	7/07/16	10/07/16
Diseño de la base de datos	10/07/16	10/07/16

Ilustración 73 Gantt final



Nombre	Fecha d...	Fecha de fin
Gestion	27/06/16	29/06/16
Analisis	1/07/16	3/07/16
Diseño	2/07/16	10/07/16
Implementacion	11/07/16	5/09/16
Definir prototipos	11/07/16	12/07/16
Prototipo 1	13/07/16	16/07/16
Prototipo 2	18/07/16	21/07/16
Prototipo 3	24/07/16	26/07/16
Prototipo 4	28/07/16	2/08/16
Prototipo 5	13/08/16	15/08/16
Prototipo 6	17/08/16	17/08/16
Prototipo 7	18/08/16	19/08/16
Prototipo 8	21/08/16	28/08/16
Prototipo 9	1/09/16	3/09/16
Prototipo 10	5/09/16	5/09/16
Plan de pruebas	11/07/16	7/09/16
Elaborar plan de pruebas	11/07/16	13/07/16
Corregir errores prototipo 1	17/07/16	17/07/16
Corregir errores prototipo 2	22/07/16	22/07/16
Corregir errores prototipo 3	27/07/16	27/07/16
Corregir errores prototipo 4	10/08/16	12/08/16
Corregir errores prototipo 5	16/08/16	16/08/16
Corregir errores prototipo 6	17/08/16	17/08/16
Corregir errores prototipo 7	20/08/16	20/08/16
Corregir errores prototipo 8	29/08/16	31/08/16
Corregir errores prototipo 9	4/09/16	4/09/16
Corregir errores prototipo 10	5/09/16	5/09/16
Prueba total	6/09/16	7/09/16

Ilustración 74 Gantt final 2



Nombre	Fecha d...	Fecha de fin
Gestion	27/06/16	29/06/16
Analisis	1/07/16	3/07/16
Diseño	2/07/16	10/07/16
Implementacion	11/07/16	5/09/16
Plan de pruebas	11/07/16	7/09/16
Documentacion	27/06/16	20/09/16
Empezar memoria	27/06/16	29/06/16
Primeros apartados	1/07/16	3/07/16
Primeros 3 puntos de la docu...	11/07/16	15/07/16
Primeros 4 puntos terminado...	25/07/16	31/07/16
Apartado 5 de la documenta...	11/08/16	13/08/16
Apartados 6 y 7	22/08/16	25/08/16
Casos de uso extendidos	29/08/16	4/09/16
Apartados 8 y 9	9/09/16	13/09/16
Preparar la defensa	14/09/16	20/09/16

Ilustración 75 Gantt final 3

9.2 Futuro para KuroObi

Como se ha mencionado en apartados anteriores, este TFG no está diseñado para ser olvidado, sino que será subido a un servidor para ser lanzado al público y con suerte obtener algún cliente para poder obtener beneficios.

Debido a esto es muy posible que la página se tenga que adaptar a diferentes deportes para poder venderse como una suite deportiva. Esto implicaría nuevas funcionalidades o readaptar las que existen para que se ajusten más a los clientes. Por ejemplo, la realización de un examen no tiene sentido para un futbolista, pero una página donde guardar sus estadísticas puede ser interesante.

Centrándonos en el Karate y su aplicación es posible que se necesiten nuevas funcionalidades puesto que los requisitos los he establecido yo mismo en mi experiencia como profesor. Otros profesores o usuarios puede que echen de menos alguna funcionalidad que no he necesitado para elaborar el TFG. También es posible que algunas de las que ya estén implementadas necesiten una revisión para adaptarse a mayor cantidad de usuarios.

9.3 Reflexión personal

Me gustaría escribir este apartado desde una posición más personal donde pueda usar un lenguaje más propio de mi persona para poder realizar esta reflexión.

Este es el último apartado que escribo del trabajo junto con los agradecimientos y significan el final del TFG, pero también significan el final de una etapa. Una etapa larga, quizá demasiado, pero donde he aprendido mucho, he forjado relaciones personales muy fuertes y me he introducido en el mundo laboral a través de las prácticas.

Ha sido aquí en las prácticas donde he confirmado que el sector en el que me gustaría trabajar que es el de programación web. Es un sector que ya me venía gustando pero que lamentablemente la universidad apuesta poco por él, solo la asignatura optativa de “Dawe” está dedicada totalmente a ello, pero que perfectamente podría ser una asignatura básica ya que se requiere mucho en el mundo laboral.

Cuando acepté las prácticas entré muy motivado y con ganas de aprender para poder aplicar todo ese conocimiento en el TFG. Para mí el TFG ha sido un trabajo de lo más estimulante puesto que me ha permitido fusionar el karate, que llevo practicando desde los nueve años, y la programación web e incluso poder crear mi propio dominio y alojar el TFG para que esté accesible a todo el mundo. Una experiencia increíble pues nunca pensaba que fuera a tener mi propia web.

Como digo un camino largo pero muy interesante y con un futuro ahora por descubrir con grandes planes. Espero que con un poco de suerte pueda trabajar en desarrollo web, cosa que no sé, puesto que el mercado laboral manda y hay que adaptarse, y poder aprender más y más sobre este mundo del que me queda muchísimo por aprender aún.

10 Agradecimientos

En este apartado me gustaría agradecer personalmente a todas aquellas personas que han servido de ayuda, dando su apoyo y opinión para la mejora de este trabajo.

“A mi hermana Noelia muchas gracias por las opiniones que me has dado, por tu punto de vista y por estar siempre viniendo a mi habitación cada vez que avanzaba 2 líneas en el código y quería saber tu opinión, gracias.”

“A Miren sobre todo por tu apoyo incondicional no solo en el TFG sino en este tramo final de la carrera. Por aguantarme durante la creación de la web que tuvo buenos y malos momentos, pero siempre estabas ahí para darme ánimos. También, aunque no me guste decirlo, gracias por tener la idea de “Kuro-Obi” nombre que encaja perfectamente con lo que es la página y el proyecto representa, gracias.”

“A mi tutor Juanan, por esos grandes ratos que habrás pasado corrigiendo las faltas de mi trabajo, por tu disposición siempre que te he requerido para cualquier cosa y por tus consejos sobre cómo hacer una buena documentación. Pero sobre todo por darme la oportunidad de hacer el trabajo contigo, gracias.”

“Gracias a Alfonso de Shkways, por darme tu opinión en todo momento siempre de manera constructiva, aunque no estuviéramos de acuerdo en algunas ocasiones, muchas gracias por la oportunidad de formar parte de la empresa y por todo el conocimiento que he adquirido trabajando ahí pues ha sido vital para el TFG y me ha simplificado mucho la vida, gracias.”

Y, por último, muchas gracias a ti lector por dedicar un buen rato a leer esta documentación ya sea para evaluarla, para guiarte o ayudarte en la elaboración de tu propia documentación.

Muchas gracias a todos.

Jonathan Guijarro.

11 Bibliografía

Referencia web

Todos los recursos usados han sido por medio de soporte digital.

Cliente/servidor. Wikipedia [1] Imagen [2]

https://es.wikipedia.org/wiki/Cliente-servidor

Redes. Arquitectura Cliente/Servidor en 3 niveles [3]

http://es.ccm.net/contents/147-redes-arquitectura-cliente-servidor-en-3-niveles

PHP. Wikipedia [4]

https://es.wikipedia.org/wiki/PHP

HTML. Wikipedia [5]

https://es.wikipedia.org/wiki/HTML

Manual de CodeIgniter. DesarrolloWeb

http://www.desarrolloweb.com/manuales/manual-codeigniter.html

Tutorial CodeIgniter. Adwe

http://www.adwe.es/codigo/tutorial-codeigniter-1o-parte-modelo-mvc-y-primeros-pasos

Introducción a CodeIgniter. Libro Digital UDEMY

https://www.udemy.com/introduccion-codeigniter/

Anexo I

Casos de uso extendidos

Anexo I – Casos de uso extendidos

En este apartado se van a definir de manera detallada algunos de los casos de uso del modelo de casos de uso especificado en el apartado de captura de requisitos. Esto sirve para detallar en profundidad todas las funcionalidades que se quieren implementar.

Se explicarán los casos de uso extendidos más relevantes o que implican mayor complejidad. El resto de casos de uso se podrán ver siguiendo este enlace.

https://dl.dropboxusercontent.com/u/13600207/proyecto/Mas%20casos%20de%20uso%20extendidos.pdf

Comenzaremos con los casos de uso del profesor, aunque pueda darse el caso que sea el mismo para el alumno como es el caso de "identificación". Se mostrarán las interfaces obtenidas por la versión móvil de la web. Para este apartado se ha usado un usuario profesor llamado "Profesores" y un usuario alumno llamado "Alumno"

1. Control de asistencia

Nombre: Controlar asistencia.
Descripción: El profesor podrá controlar la asistencia de los alumnos mediante "Swipe" o deslizamiento. Si desliza a la izquierda marcara que el alumno no está. Si lo hace a la derecha lo marcara como presente.
Actores: Profesor.
Precondiciones: Estar identificado como profesor.
Tener alumnos registrados en el sistema.
Requisitos funcionales: Ninguno.
Flujo de eventos:

1. Si el profesor desliza a la izquierda se marca al alumno como ausente y se muestra un mensaje (ilustración 76).
2. Si el profesor desliza a la derecha se marca al alumno como presente y se muestra un mensaje (ilustración 77).

Postcondiciones: Se guarda la asistencia del alumno y se incluye en la tabla de asistencia marcando el día en verde si esta y en rojo está ausente.
Interfaz gráfica:

Ilustración 76 Asistencia alumno ausente

Ilustración 77 Asistencia alumno presente

2. Realizar Examen

Nombre: Realizar examen.
Descripción: El profesor puede realizar exámenes a los alumnos de un grupo.
Actores: Profesor.
Precondiciones: Estar identificado como profesor.
Tener alumnos dados de alta en el sistema.
Requisitos funcionales: Ninguno.
Flujo de eventos:

1. El profesor pulsa “iniciar examen” (ilustración 78).

2. En la pantalla se muestra una tabla de los alumnos del grupo y se muestra si han realizado el examen. En caso afirmativo se muestra la nota. En caso de no haber realizado el examen no se muestra nada. El profesor pulsa en "iniciar examen" (Ilustración 79).
3. Se muestra la interface del examen (ilustración 80 y 81).
    - **3.1.** El profesor mueve los círculos de puntuación para establecer la nota de ese apartado.
    - **3.2.** El profesor puede añadir comentarios al examen (ilustración 82).
        - **3.2.1.** El profesor pulsa en "seleccionar archivo" lo que activa la cámara del dispositivo móvil para iniciar la grabación del vídeo. Una vez grabado vuelve a la interface del examen (ilustración 83).
        - **3.2.2.** El profesor puede subir el examen a *YouTube* pulsando sobre "subir vídeo" lo que inicia la subida del vídeo (ilustración 84).
        - **3.2.3.** Una vez el vídeo esta subido, se copia el enlace en el cuadro "id del vídeo".
        - **3.2.4.** El profesor pulsa "guardar examen".
            - **3.2.4.1.** Si hay más alumnos se carga el siguiente alumno.
            - **3.2.4.2.** Si no hay más alumnos se muestra un mensaje por pantalla (ilustración 85).

Postcondiciones: El examen se guarda en el sistema y se muestra en la tabla de exámenes.

Interfaz gráfica:

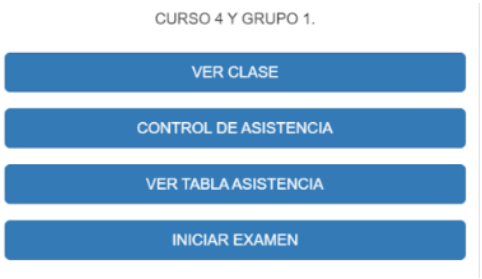


Ilustración 78 Iniciar examen

Ilustración 79 Tabla examen

Ilustración 80 Interface examen I

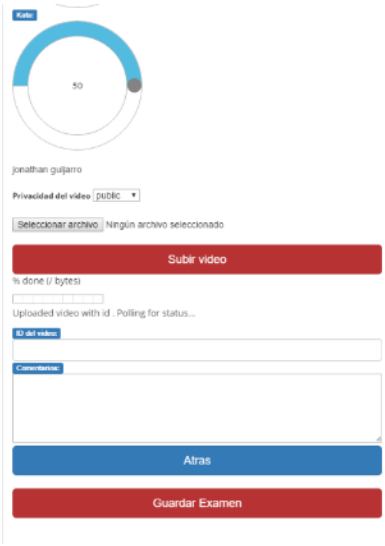


Ilustración 81 Interface examen II

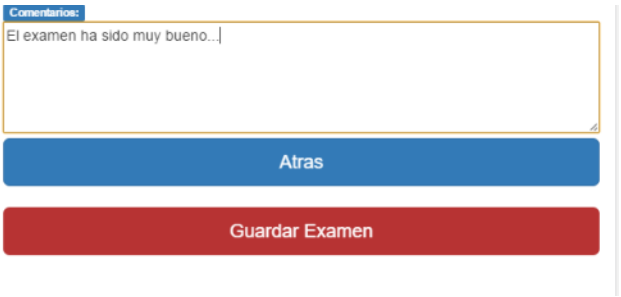


Ilustración 82 Comentario examen

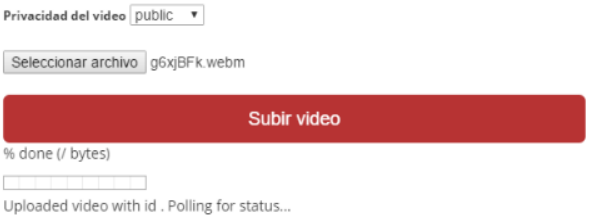


Ilustración 83 Selección de archivo

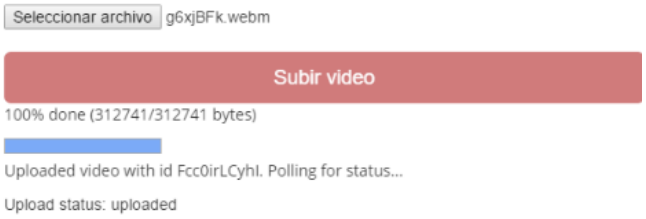


Ilustración 84 Progreso vídeo

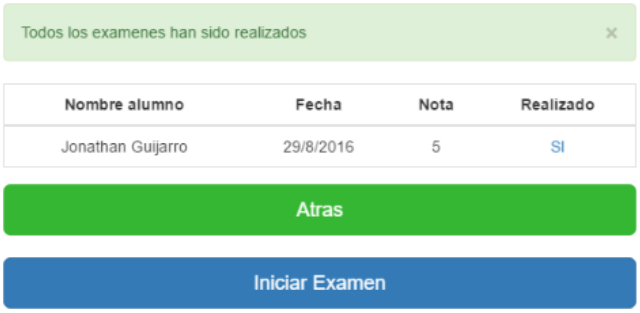


Ilustración 85 Exámenes realizados

3. Crear wazas

Nombre: Gestionar wazas
Descripción: El alumno puede crear carruseles con los movimientos de las *Wazas* para más tarde subirlo a su *google drive* personal. Puede notificar al profesor para que vea la *Waza*.
Actores: Usuario alumno.
Precondiciones: Estar identificado en el sistema como alumno.
Requisitos funcionales: Ninguno.
Flujo de eventos:

1. El alumno pulsa sobre "crear wazas" en la página principal (ilustración 86).
2. En menú con las distintas wazas escoge la que quiera crear (ilustración 87),
    - **2.1.** En la pantalla se muestran los campos para sacar las 6 fotos y crear el carrusel (ilustración 88).
        - **2.1.1.** Si el alumno deja algún campo por rellenar se mostrará el aviso (ilustración 89)
        - **2.1.2.** Una vez tenga todos los campos rellenos pulsa "aceptar"
        - **2.1.3.** Se mostrará el carrusel formado y podrá elegir que foto subir a su google drive personal con el botón que aparece debajo de cada foto (ilustración 90).

Postcondiciones: Se guarda la *Waza* en el sistema y se crea un recordatorio en la página principal (ilustración 91).
Interfaz gráfica:

Ilustración 86 Gestionar wazas

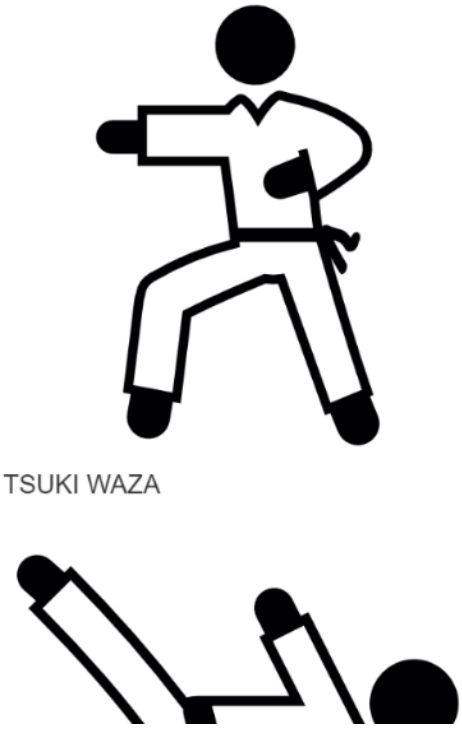


Ilustración 87 Elegir waza

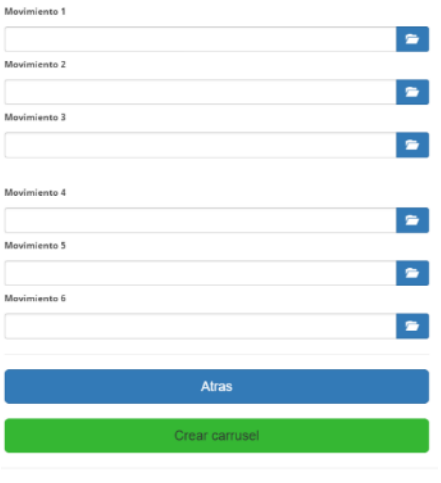


Ilustración 88 Crear wazas fotos

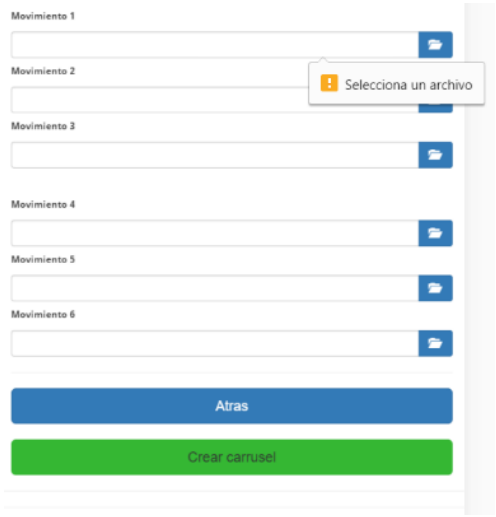


Ilustración 89 Mensaje crear wazas

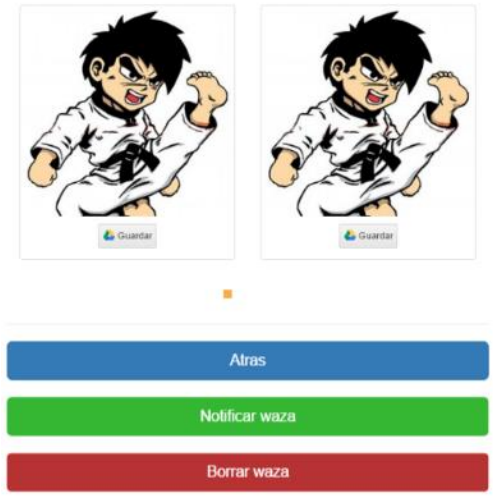


Ilustración 90 Ver waza

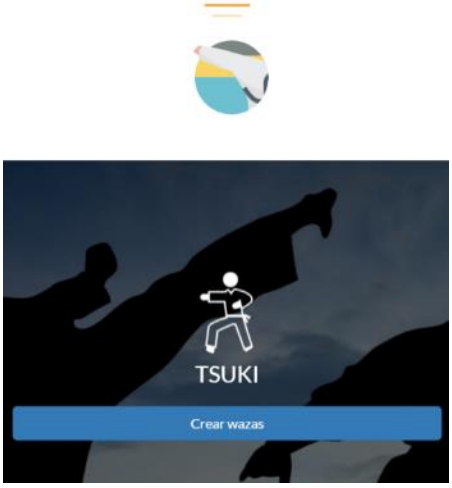


Ilustración 91 Icono principal waza

Anexo II
Diagramas de secuencia

Anexo II - Diagramas de secuencia

En este apartado se incluirán algunos de los diagramas de secuencia realizados. Estos diagramas muestran como interaccionan los actores con el sistema y como el sistema se comunica internamente con las diferentes interfaces, funciones y bases de datos.

Al igual que el apartado anterior se han incluido solo los más relevantes o interesantes el resto se podrán encontrar en el siguiente enlace:

https://dl.dropboxusercontent.com/u/13600207/proyecto/Mas%20diagramas%20de%20secuencia.pdf

También muestran los efectos que producen dichas operaciones, desde cambios en la base de datos hasta salidas que producen el sistema cuando se invocan dichas funciones. Los diagramas muestran los mensajes en orden cronológico siguiendo el curso según se producen de arriba abajo.

1. Controlar asistencia

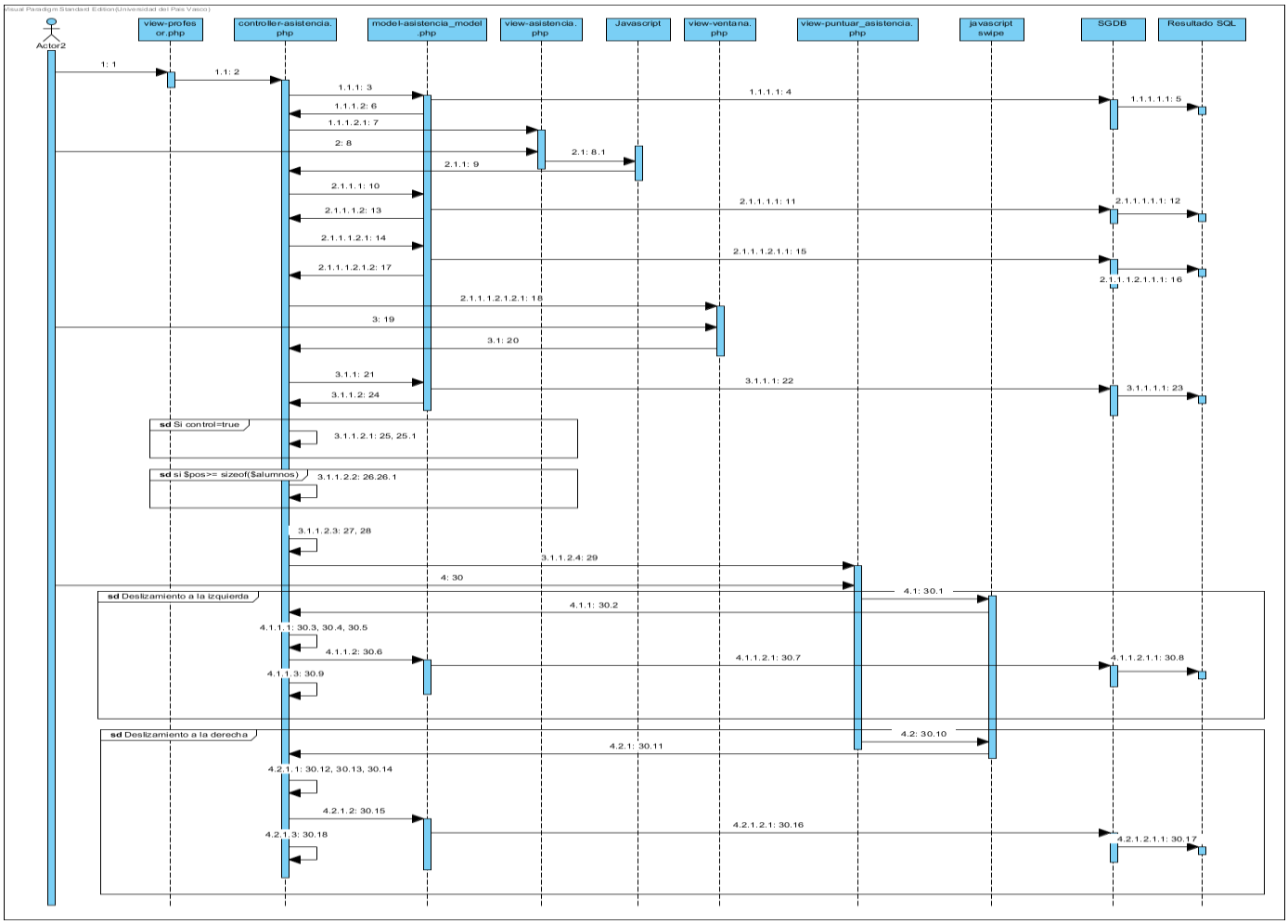


Ilustración 92 D.S Control asistencia

1. Usuario identificado como profesor se encuentra en su página principal "profesor.php"
   El profesor pulsa sobre "gestión de alumno".
2. href="index.php?/asistencia/asistencias".
   Se carga la función asistencias() y llama al modelo "asistencias_model.php" para cargar todos los colegios del profesor.
3. $data['colegios']=$this->asistencia_model->colegios($this->session->userdata('ID'));
4. $sql= "SELECT DISTINCT `colegio` FROM `kt_alumnos` WHERE `ID_profesor` = $ID";
5. $query=$this->db->query($sql)->result();
6. return $query;
   Cargamos la interface "asistencia.php" y le enviamos $data.
7. $this->load->view('asistencia',$data);
   El profesor selecciona el colegio en el desplegable. Y con mediante JavaScript carga dinámicamente los grupos y cursos de ese colegio.
8. function showUser(str){…
   la función llama a otra función "asistencia_alumno()" enviándole el colegio elegido y se encarga de cargar los grupos y cursos.
9. xmlhttp.open("GET","index.php?/asistencia/asistencia_alumno/"+str,true);
   Cargamos de nuevo el colegio de la base de datos y lo guardamos en $data
10. $data['colegio']=$this->asistencia_model->get_colegio($sp);
11. $sql= "SELECT * FROM `kt_alumnos` WHERE `Colegio` = '".$colegio."'";
12. $query=$this->db->query($sql)->result();
13. return $query;
    Cargamos los cursos de ese colegio y sus grupos.
14. $data['cursos']=$this->asistencia_model->grupos($sp);
15. $sql="SELECT DISTINCT `curso`,`grupo` FROM `kt_alumnos` WHERE `Colegio` ='".$colegio."'";
16. $query=$this->db->query($sql)->result();
17. return $query;
    Enviamos todo a la vista "ventana.php" que se cargará dentro de la vista "asistencia.php"
18. $this->load->view('ventana',$data);
    Se mostrarán los diferentes cursos y grupos que hay dentro del colegio elegido.
19. El profesor pulsa sobre "control de asistencia"
20. href="index.php?/asistencia/puntuar/<?=$total?>"
    Se carga la función "puntua($total,$pos=0,$control=false)" por defecto la variable control está a false si no le llega nada y $pos es 0.
    Se cargan todos los alumnos de ese grupo
21. $alumnos=$this->asistencia_model->get_alumnos($separados[0],$separados[1],$separados[2]);
22. $sql="SELECT * FROM `kt_alumnos` WHERE `colegio`='".$colegio."' AND `Curso`= $curso AND`Grupo`= $grupo";
23. $query=$this->db->query($sql)->result();
24. return $query;
25. ***[Si control es true]***

25.1. $pos=$pos+1;

[fin si]

Se comprueba la posición para evitar salidas de array

26. [Si $pos >= sizeof($alumnos)]

Comprobación completa

26.1. redirect('../index.php?/asistencia/asistencias', 'refresh');

[fin si]

Cargamos $pos en $data

27. $data['pos']=$pos;

Cargamos la array $alumnos en $data

28. $data['alumnos']=$alumnos;

Lo enviamos a la vista “puntuar_asistencia.php” junto con $data.

29. $this->load->view('puntuar_asistencia',$data);

30. Se muestra la imagen del alumno, el profesor controlara la asistencia del alumno deslizando a derecha o izquierda sobre la foto.

[Si el profesor desliza hacia la izquierda (alumno ausente) {

Se carga la función en JavaScript correspondiente

30.1. $(document).on("pagecreate","#pageone",function(){
$("img").on("swipeleft",function(){ $("span").text("Alumno ausente");
…

30.2. window.location="index.php?/asistencia/no_esta/"+colegio+"-"+curso+"-"+grupo+"-"+pos+"-"+id;

Vamos a la función “no_esta()” en “asistencia.php”

Cogemos la fecha de hoy y la tratamos para separarla en día, mes y año

30.3. $date=getdate();$día=$date['mday'];
$mes=$date['mon'];$año=$date['year'];

30.4. $control=true;

30.5. $asist='No';

Grabamos la asistencia en la base de datos.

30.6. $this->asistencia_model->marcar_asistencia($id_alumno,$dia,$mes,$año,$asist);

30.7. $sql="INSERT INTO `kt_asistencia`(`ID_alumno`, `Dia`, `Mes`, `Año`,`Asistencia`) VALUES ($id_alumno, $día, $mes, $año, '".$asist."')";

30.8. $this->db->query($sql);

Se vuelve el control a “puntuar()” para el siguiente alumno.

30.9. $this->puntuar($sp,$separados[3],$control);

[Si el profesor desliza hacia la derecha (alumno presente)]

Se carga la función en JavaScript correspondiente.

30.10. $(document).on("pagecreate","#pageone",function(){
$("img").on("swiperight",function(){ $("span").text("Alumno en clase");
…

30.11. window.location="index.php?/asistencia/no_esta/"+colegio+"-"+curso+"-"+grupo+"-"+pos+"-"+id;

Vamos a la función “esta()” en “asistencia.php”

Cogemos la fecha de hoy y la tratamos para separarla en día, mes y año

30.12. $date=getdate();$dia=$date['mday'];$mes=$date['mon'];$año=$date['year'];
30.13. $control=true;
30.14. $asist='Si';

Grabamos la asistencia en la base de datos.

30.15. $this->asistencia_model->marcar_asistencia($id_alumno,$dia,$mes,$año,$asist);
30.16. $sql="INSERT INTO `kt_asistencia`(`ID_alumno`, `Dia`, `Mes`, `Año`,`Asistencia`) VALUES ($id_alumno, $día, $mes, $año, '".$asist."')";
30.17. $this->db->query($sql);

Se vuelve el control a “puntuar()” para el siguiente alumno.

30.18. $this->puntuar($sp,$separados[3],$control);

2. Realizar examen

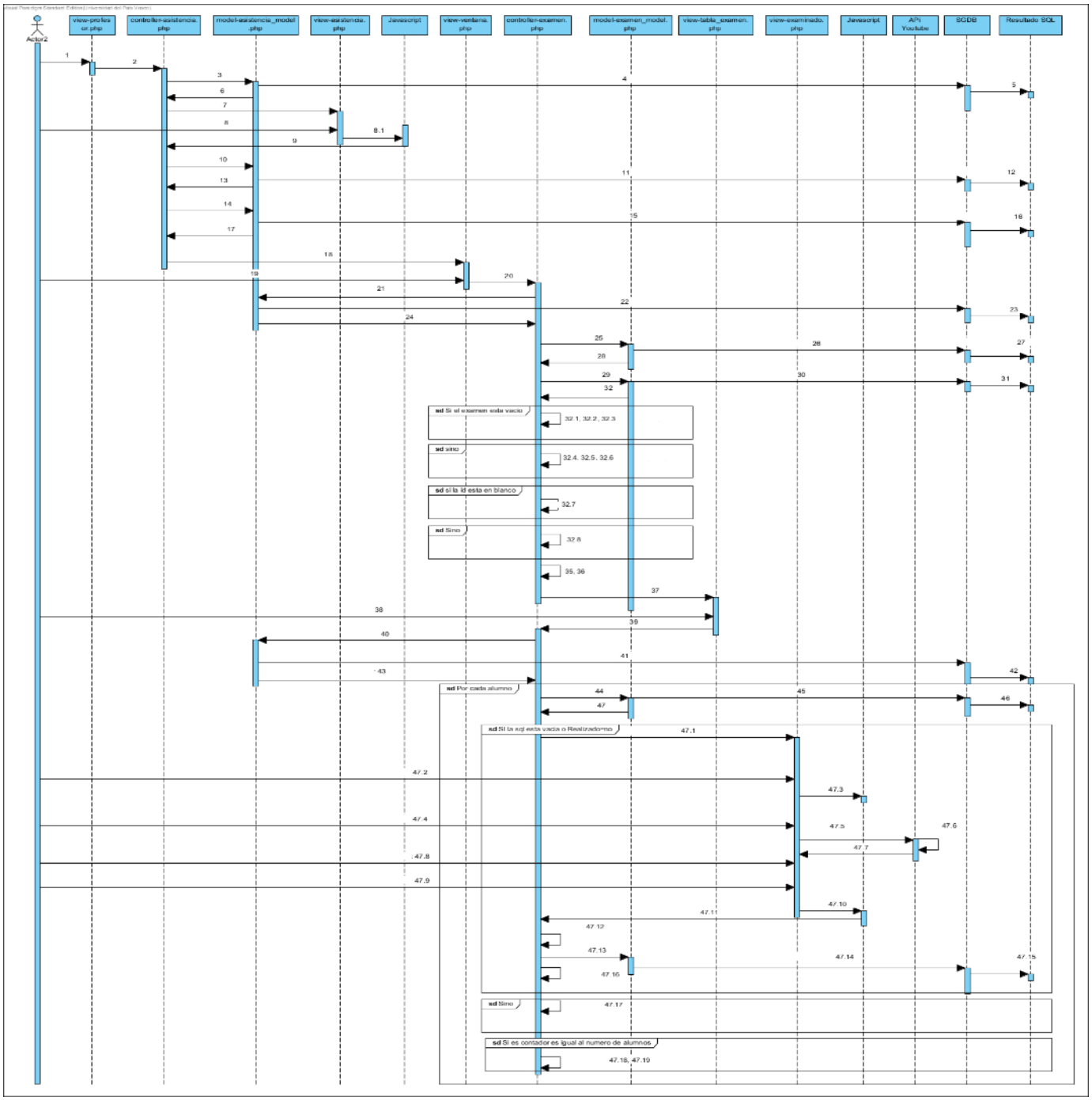


Ilustración 93 D.S. Realizar examen

1. Usuario identificado como profesor se encuentra en su página principal “profesor.php”
   El profesor pulsa sobre “gestión de alumno”.
2. href="index.php?/asistencia/asistencias".
   Se carga la función asistencias() y llama al modelo “asistencias_model.php” para cargar todos los colegios del profesor.

3. $data['colegios']=$this->asistencia_model->colegios($this->session->userdata('ID'));
4. $sql= "SELECT DISTINCT `colegio` FROM `kt_alumnos` WHERE `ID_profesor` = $ID";
5. $query=$this->db->query($sql)->result();
6. return $query;

Cargamos la interface “asistencia.php” y le enviamos $data.

7. $this->load->view('asistencia',$data);

El profesor selecciona el colegio en el desplegable. Y con mediante JavaScript carga dinámicamente los grupos y cursos de ese colegio.

8. function showUser(str){…

La función llama a otra función “asistencia_alumno()” enviándole el colegio elegido y se encarga de cargar los grupos y cursos.

9. xmlhttp.open("GET","index.php?/asistencia/asistencia_alumno/"+str,true);

Cargamos de nuevo el colegio de la base de datos y lo guardamos en $data

10. $data['colegio']=$this->asistencia_model->get_colegio($sp);
11. $sql= "SELECT * FROM `kt_alumnos` WHERE `Colegio` = '".$colegio."'";
12. $query=$this->db->query($sql)->result();
13. return $query;

Cargamos los cursos de ese colegio y sus grupos.

14. $data['cursos']=$this->asistencia_model->grupos($sp);
15. $sql="SELECT DISTINCT `curso`,`grupo` FROM `kt_alumnos` WHERE `Colegio` ='".$colegio."'";
16. $query=$this->db->query($sql)->result();
17. return $query;

Enviamos todo a la vista “ventana.php” que se cargará dentro de la vista “asistencia.php”

18. $this->load->view('ventana',$data);

Se mostrarán los diferentes cursos y grupos que hay dentro del colegio elegido.

19. El profesor pulsa sobre “Iniciar examen”
20. href="index.php?/examen/ver/<?=$total?>

Coge todos los alumnos del profesor

21. $alumnos=$this->asistencia_model->get_alumnos($separados[0],$separados[1],$separados[2]);
22. $sql="SELECT * FROM `kt_alumnos` WHERE `colegio`='".$colegio."' AND `Curso`= $curso AND`Grupo`= $grupo";
23. $query=$this->db->query($sql)->result();
24. return $query;

[Por cada alumno]

Se mira en la base datos si ha realizado el examen.

25. $res=$this->examen_model->get_alumno_exam($alum->ID);
26. $this->db->select('*');$this->db->from('kt_examen ktex');$this->db->join('kt_alumnos ktal', 'ktex.ID_alumno=ktal.ID'); $this->db->where('ktal.ID', $ID);
27. $query = $this->db->get();
28. return $query->result();

Se coge la id del examen del alumno.

29. $id_examen=$this->examen_model->get_exam_id($alum->ID);
30. $this->db->select('ID');$this->db->from('kt_examen');$this->db->where('ID_alumno', $ID);
31. $query = $this->db->get();
32. return $query->result();

[Si el examen esta vacío]

32.1. $fecha="/";
32.2. $nota="/";
32.3. $realizado="NO";

[sino]

32.4. $fecha=$res[0]->Fecha;
32.5. $nota=$res[0]->Nota;
32.6. $realizado=$res[0]->Realizado;

[fin si]

[Si la id del examen esta en blanco]

32.7. $alumno_examen[] = array('ID_alumno' => $alum->ID, 'Nombre'=>$alum->Nombre.' '.$alum->Apellidos, 'Fecha'=>$fecha, 'Nota'=>$nota, 'Realizado'=>$realizado,'ID'=>null);

[sino]

32.8. $alumno_examen[] = array('ID_alumno' => $alum->ID,'Nombre'=>$alum->Nombre.' '.$alum->Apellidos, 'Fecha'=>$fecha ,'Nota'=>$nota, 'Realizado'=>$realizado, 'ID'=>$id_examen[0]->ID);

33. [fin si]

34. [fin por cada]

Metemos todo en $data

35. $data['alumnos']=$total;
36. $data['alumnos_examen']=$alumno_examen;

Cargamos la interface “tabla_examen.php”

37. $this->load->view('tabla_examen',$data);
38. El profesor pulsa en el botón “iniciar examen” que iniciara la tanda de exámenes para todos los alumnos. Cargándose la función “examen_start()” en “examen.php”
39. href="index.php?/examen/examen_start/<?=$alumnos?>

Se cargan todos los alumnos del profesor

40. $alumnos=$this->asistencia_model->get_alumnos($separados[0],$separados[1],$separados[2]);
41. $sql="SELECT * FROM `kt_alumnos` WHERE `colegio`='".$colegio."' AND `Curso`= $curso AND`Grupo`= $grupo";
42. $query=$this->db->query($sql)->result();
43. return $query;

[Por cada alumno]

Se carga el examen del alumno

44. $res=$this->examen_model->get_alumno_exam($alum->ID);

45. $this->db->select('*');$this->db->from('kt_examen ktex');$this->db->join('kt_alumnos ktal', 'ktex.ID_alumno=ktal.ID'); $this->db->where('ktal.ID', $ID);
46. $query = $this->db->get();
47. return $query->result();

[Si la secuencia está vacía o Realizado == No]

Carga la vista examinado que contiene el examen a rellenar.

47.1. $this->load->view('examinado',$data);

47.2. El profesor rellena los campos.
Si pulsa un círculo de puntuación se llama a la función de JavaScript que actualiza el valor del círculo.

47.3. roundSlider()

47.4. El usuario pulsa “upload vídeo”

47.5. Se conecta con la API de YouTube

47.6. Se sube el vídeo

47.7. Se obtiene el enlace en la pantalla principal

47.8. El profesor copia el enlace en la caja de texto

47.9. Pulsa “guardar examen”

47.10. Se llama a la función JavaScript ejecutar () que comprueba todos los campos y llama a función “puntuar_examen” en “examen.php” pasándole todas las variables.

47.11. window.location="index.php?/examen/puntuar_examen/"+id+"-"+t+"-"+k+"-"+c+"-"+p+"-"+z+"-"+total+"-"+v;

47.12. Se procesan las variables.
Se llama al “examen_model.php” para que dé de alta en la base de datos los resultados del examen.

47.13. $this->examen_model->puntuar_examen($separados[0],$separados[1],$separados[2],$separados[3],$separados[4],$comentarios,$fecha,$id_profesor,$nota_final,$video);

47.14. $sql="UPDATE `kt_examen` SET `ID_alumno`=$id,`ID_profesor`=$id_profesor,`Tecnica`=$tecnica,`Kata`=$kata,`Combinaciones`=$combinaciones,`Posiciones`=$posiciones,`Comentarios`='".$comentarios."',`Nota`=$nota_final,`Video`='".$video."',`Realizado`='SI',`Fecha`='".$dia."' WHERE `ID`=$id_examen";

47.15. $this->db->query($sql);
Se re-direcciona a la función “examen_start() para que pase al siguiente alumno

47.16. redirect('../index.php?/examen/examen_start/'.$total, 'refresh');

[sino]

Aumenta el contador

47.17. $cont++;

[fin si]

[Si el contador es igual al número total de alumnos]

Cargamos la interface ver (tabla del examen) con un mensaje.

47.18. $this->session->set_flashdata('mensaje', 'Todos los examenes han sido realizados');

47.19. redirect('../index.php?/examen/ver/'.$total, 'refresh');

[fin si]

[fin por cada]

3. Grabar sesiones

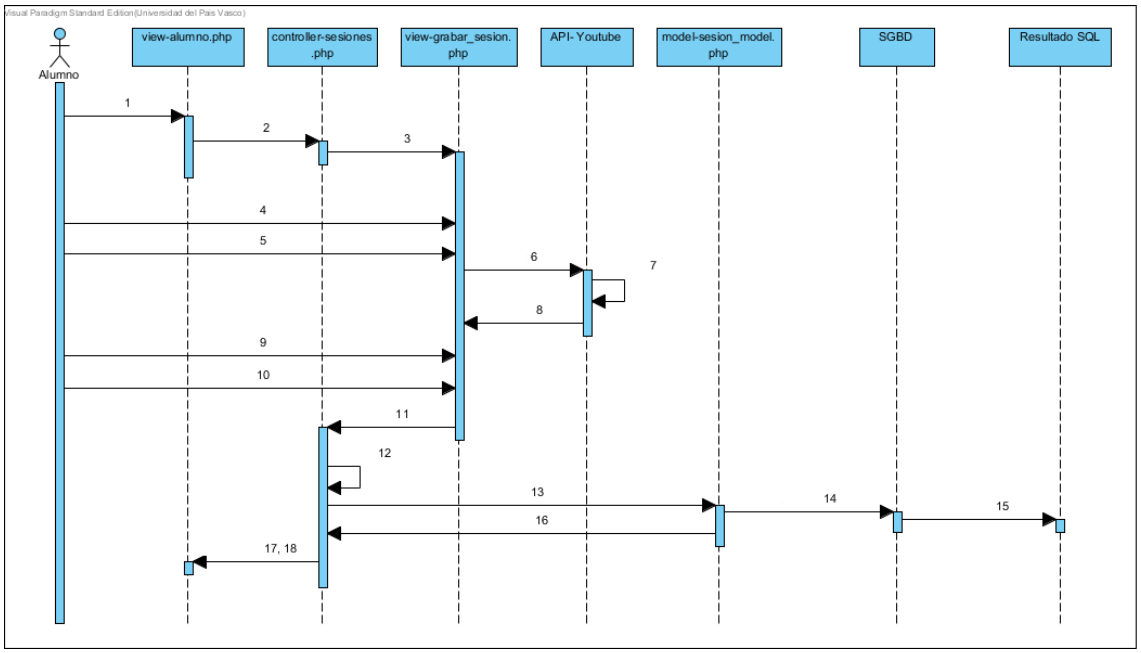


Ilustración 94 D. S Grabar sesiones

1. Usuario identificado como alumno se encuentra en su página principal "alumno.php"

El alumno pulsa sobre "gestión de entrenamientos".

2. href="index.php?/sesiones/grabar_sesion".
   Cargamos la vista "grabar_sesion.php".
3. $this->load->view('grabar_sesion');
4. El alumno graba el vídeo y rellena los campos.
5. El usuario pulsa "upload vídeo"
6. Se conecta con la API de YouTube
7. Se sube el vídeo
8. Se obtiene el enlace en la pantalla principal
9. El alumno copia el enlace en la caja de texto
10. Pulsa subir sesión
    Se carga la función grabar()
11. action="index.php?/sesiones/grabar"
12. Se coge la fecha actual del sistema.
    Se conecta con el modelo para que lo registre en la base de datos.
13. $this->sesion_model->grabar($id,$id_video,$fecha,$titulo_video)
14. $sql = "INSERT INTO `kt_sesion`(`ID_alumno`, `Fecha`, `enlace`,`titulo`) VALUES ($id,'".$fecha."','".$id_video."','".$titulo_video."')";
15. $query=$this->db->query($sql);
16. return $query;
    17Se redirección a la página principal y se muestra un mensaje.
17. $this->session->set_flashdata('mensaje3', 'Sesión grabada correctamente');

18. redirect('../index.php?/welcome/', 'refresh');

4. Crear waza

NOTA*: El usuario tiene a disposición 4 tipos de wazas, la secuencia es la misma solo que cambia las pantallas intermedias y las funciones que las llaman. Se realizará la secuencia para una de ellas para no alargar el diagrama 3 veces con elementos similares.

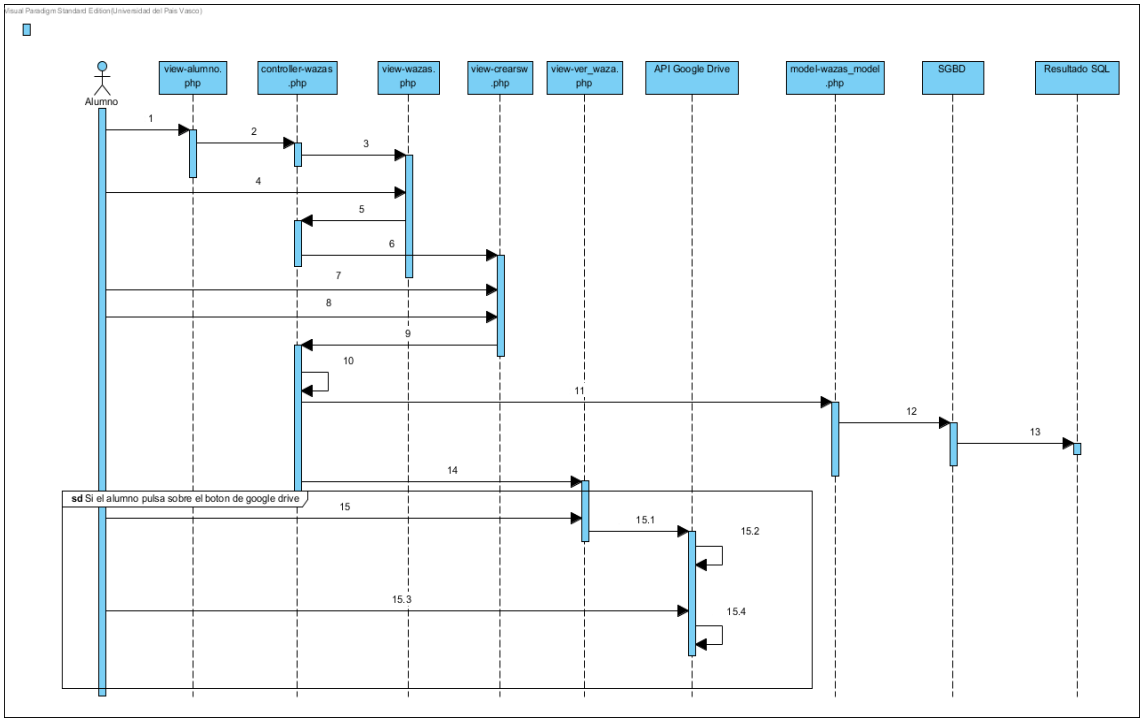


Ilustración 95 D.S Crear waza

1. Usuario identificado como alumno se encuentra en su página principal "alumno.php"
   El alumno pulsa sobre "gestión de entrenamientos".
2. href="index.php?/wazas/gestionar_wazas".
   Cargamos la vista "wazas.php"
3. $this->load->view('wazas');
4. El alumno elige el tipo de waza que va a crear y pulsa sobre la imagen.
5. href="index.php?/wazas/crear_waza/1"
   Se carga la vista para crear el waza seleccionado.
6. $this->load->view('crear_sukiw');
7. El alumno saca las 6 fotos necesarias para la creación del waza.
8. El alumno pulsa "crear carrusel".
9. action="index.php?/wazas/new_sukiw"
10. Se procesan las imágenes que se suben al servidor.
    Se sube la ruta de las imágenes a la base de datos.
11. $this->wazas_model->crear_waza($id_alumno,$ruta_imagen1,$ruta_imagen2,$ruta_imagen3,$ruta_imagen4,$ruta_imagen5,$ruta_imagen6,$tipo_waza);

12. $sql="INSERT INTO `kt_wazas`(`ID_alumno`, `Ruta1`, `Ruta2`, `Ruta3`, `Ruta4`, `Ruta5`, `Ruta6`, `Tipo_waza`) VALUES ($id_alumno,'".$ruta_imagen1."','".$ruta_imagen2."','".$ruta_imagen3."','".$ruta_imagen4."','".$ruta_imagen5."','".$ruta_imagen6."','".$tipo_waza."')";
13. $this->db->query($sql);
    Se carga la vista “ver_waza.php” donde está el carrusel creado con las fotos.
14. $this->load->view('ver_waza',$data);
15. ***[Si el alumno pulsa sobre el botón de google drive]***
    15.1. Se conecta con la API de Google Drive.
    15.2. Se pide autorización a google.
    15.3. Se selecciona la carpeta
    15.4. Se sube la foto
    [fin si]