Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación web para la formación de equipos de trabajo

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jose Vicente Bustos Ferrer

Tutores: Vicente Javier Julián Inglada, Juan Miguel Alberola Oltra, Elena del Val Noguera

2015-2016

Resumen

El objetivo del proyecto será desarrollar una aplicación web orientada a la formación de grupos de trabajo. El sistema tendrá que permitir la gestión de usuarios, gestión de actividades grupales, encuestas y formularios de evaluación. El sistema interactuará con una base de datos que habrá que desarrollar y unos algoritmos de formación de equipos que ya están implementados.

Palabras clave: aplicación web, equipos, gestión, usuarios, actividades, encuestas, formularios, base de datos, algoritmo

Abstract

The aim of the project is to develop a web application oriented to formation of working groups. The system will need to allow user management, group management activities, surveys and evaluation forms. The system will interact with a database to be developed and teams' formation algorithms already implemented.

Keywords : web application, groups' formation, user management, surveys, forms, database, algorithm

Tabla de contenidos

1 Introducción

1.1 Contexto y motivación

En pleno comienzo del siglo XXI, Internet ha cambiado mucho desde que allá por el 1961 se escribieran los primeros artículos sobre conmutación de paquetes [1]. Cincuenta y cinco años después, su penetración en el mundo ha llegado al 51 % mundial [2], con más de tres mil millones y medio de personas conectadas a la red.

Esto no solo ha supuesto una expansión descomunal de la información que ahora es accesible desde cualquier lugar del mundo y desde cualquier dispositivo con una tarjeta de red, haciendo así posible la creación de nuevas tecnologías y herramientas, sino que también ha abierto la puerta a nuevas oportunidades de negocio. Es por ello por lo que las organizaciones y empresas necesitan renovar sus procesos de trabajo, adaptándolos a las nuevas necesidades de la sociedad, o de lo contrario podrían acabar siendo relegadas por competidores que consiguen llegar a una mayor audiencia y ofrecerle a esta un servicio más rápido y eficiente.

Dado este escenario, hoy podemos encontrarnos con un amplio surtido de servicios y comercios accesibles simplemente con tener un dispositivo conectado a Internet y un navegador, aprovechando la versatilidad de la *World Wide Web*, sin necesidad de instalar o comprar ningún software o accesorio adicional.

Imagen 1 Ejemplo de servicio web: buscador. Google, www.google.es

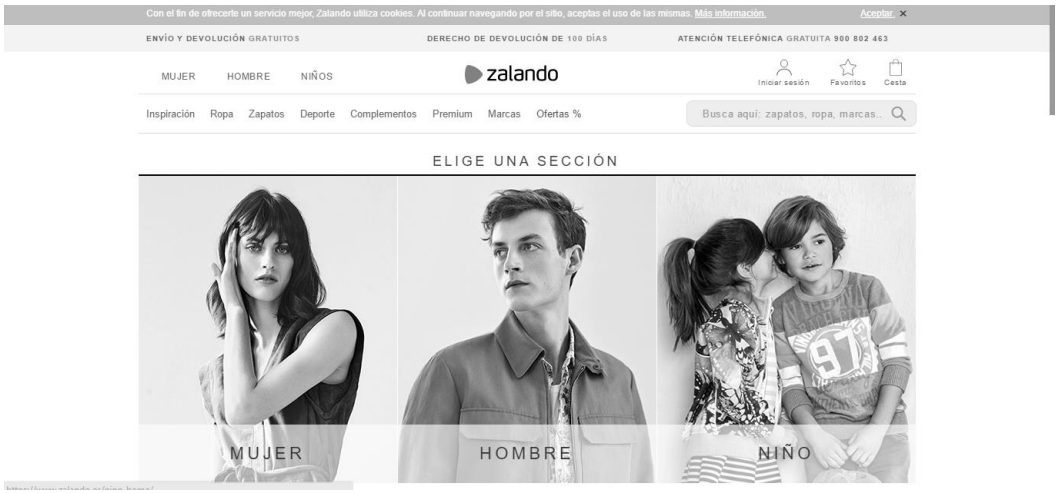


Imagen 2 Ejemplo de tienda electrónica. Zalando, www.zalando.es

Sin embargo, la sociedad no se sustenta solo de servicios y tiendas. Existen otros ámbitos en el que es necesaria alguna forma de administrar la información que se va digitalizando, ya sea agrupándola, etiquetándola… Administraciones públicas, empresas, organizaciones culturales y educativas y similares requieren organizar y cuantificar datos de miles de usuarios, estudiantes, transacciones, mensajes, estadísticas y demás información útil.

Aquí surge el concepto de aplicación web. Antes de existir la *web*, instalábamos un programa concreto en nuestro dispositivo (si era compatible), ocupando espacio de memoria y supeditado a posibles amenazas de seguridad (como la no actualización del *software*). Ahora, es posible acceder a un servicio deslocalizado, basado en estándares, cuyas actualizaciones dependenden únicamente del responsable de la aplicación y además es accesible desde cualquier sistema operativo. Por otra parte, la comunicación a través de Internet permite que puedan interactuar diferentes usuarios en el mismo sistema desde lugares distintos.

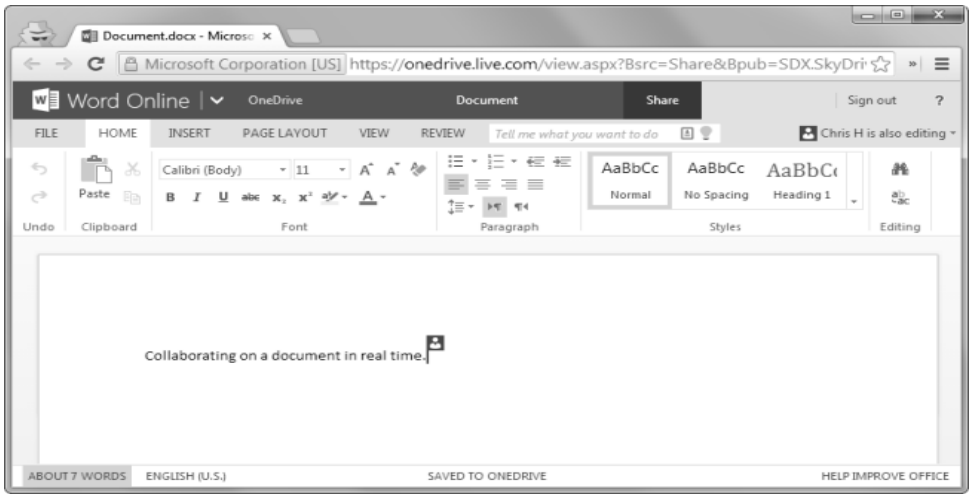


Imagen 3 Ejemplo de aplicación web: editor de documentos colaborativo. Word Online, http://www.howtogeek.com/183176/how-to-collaborate-on-documents-over-the-internet/

La **Universidad Politécnica de Valencia** (U.P.V.) y, en su extensión, el **Departamento de Sistemas Informáticos y Computación** (D.S.I.C.), en el marco de la enseñanza superior, vela constantemente por la innovación en sus métodos de docencia y de investigación, adaptándose año tras año a los constantes cambios que sufre la tecnología informática con el fin de mejorar la eficiencia y el éxito de sus programas educativos.

En el contexto de la gestión de actividades grupales en la docencia e investigación, y tras los resultados de experimentos realizados por el departamento en el campus de Gandía de la U.P.V. (**Alberola, J. M., Del Val, E., Sanchez-Anguix, V., Palomares, A., & Teruel, M. D. (2016).** ***An artificial intelligence tool for heterogeneous team formation in the classroom. Knowledge-Based Systems*****, 101, 1-14**.), se ha considerado necesaria la utilización de una herramienta que permita generar **equipos de trabajo** basados en **aspectos objetivos** de los participantes, en lugar de permitir que la elección de los integrantes del grupo dependa únicamente de su decisión personal.

Actualmente, la U.P.V. cuenta con un sistema de gestión de contenidos basado en *Sakai* [4], llamado PoliformaT, utilizado de forma multidisciplinar por toda la comunidad universitaria. Sin embargo, el mismo no dispone de suficiente versatilidad y especialización en cuanto a la gestión de actividades grupales que se requiere para llevar a cabo la correcta formación de los equipos a través de un algoritmo externo ya desarrollado. Además, no cuenta con un sistema de retroalimentación que permita un seguimiento de las actividades y del éxito de sus grupos en cuanto al objetivo de la tarea.

Por ello, se decide la creación de una nueva aplicación web especializada que de solución a la problemática planteada, aprovechando la actualización de los estándares web respecto a nuevas tecnologías y mejoras de usabilidad y enfocando el desarrollo de dicha aplicación a la formación e investigación en el ámbito de un Trabajo de Fin de Grado.

1.2 Objetivos

La aplicación web tendrá como eje fundamental la **generación automática de grupos siguiendo distintos criterios**. Además, debe gestionar adecuadamente usuarios, actividades, tests de autopercepción y encuestas de satisfacción.

El sistema interactuará con un algoritmo externo de generación de equipos que será proporcionado por el D.S.I.C. y guardará los datos obtenidos en una base de datos para posterior análisis y consulta.

Teniendo en cuenta su ámbito de uso, es necesario que la aplicación sea accesible desde dispositivos con diferentes tamaños de pantalla y que genere un consumo reducido de banda ancha, debido al auge mundial en la utilización de los dispositivos móviles:

"El 40% de las personas mira la pantalla de su teléfono más de 50 veces al día mientras que el 70% mira su teléfono durante la primera media hora después de haberse despertado"

(INFORME *DITRENDIA*: *MOBILE* EN ESPAÑA Y EN EL MUNDO, 2015:5) [5]

1.3 Tecnologías

Para la realización del proyecto deberemos repasar las tecnologías disponibles en la actualidad de forma que elijamos las que mejor se adapten a nuestras necesidades. Dado que el ámbito de la aplicación se sitúa en la *World Wide Web*, repasaremos las diferentes tecnologías web que podremos encontrar hoy en día para llevar a cabo nuestro objetivo.

Tras una profunda búsqueda en Internet y repasando asignaturas vistas en el grado en Ingeniería, podemos dar con información relevante acerca del desarrollo web [7]. En concreto, encontramos una reseña en la que repasan las tecnologías más usadas en la actualidad en el ámbito del desarrollo de aplicaciones web [8]:

1.3.1 HTTP

"*Hypertext Transfer Protocol* o *HTTP* (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la *World Wide Web*" (Wikipedia: *Hypertext Transfer Protocol*).

HTTP será el encargado de establecer la comunicación entre el servidor en el que tengamos alojada la aplicación web y el navegador del usuario. Dada la arquitectura cliente-servidor de este protocolo, deberemos enfocar el desarrollo de la aplicación a dos partes conectadas pero diferenciadas: la interfaz de la aplicación y la lógica del programa.

1.3.2 HTML

"*HTML*, siglas en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web." (Wikipedia: *HTML*)

Mediante *HTML* construiremos la interfaz de la aplicación. Basado en *XML*, permite definir la estructura de las páginas a través de marcas especificadas por el *World Wide Web Consortium* (*W3C*). En concreto, la quinta revisión de dicho lenguaje (*HTML* 5), introduce nuevas características para ayudar a desarrolladores de aplicaciones web y define unos criterios de conformidad para los agentes de usuario (navegadores web) en un esfuerzo por mejorar la interoperabilidad [6].

1.3.3 CSS

"Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en *HTML* o *XML* (y por extensión en *XHTML*)" (Wikipedia: Hoja de estilos en cascada)

Las hojas de estilo ya llevan años presentes en el mundo de la programación en la web. Su utilidad y facilidad a la hora de estilizar un sitio online las ha convertido en la herramienta indispensable de cualquier desarrollador web. Tal es su éxito, que podemos encontrar fácilmente tutoriales que ofrecen gratuitamente en Internet para cualquier nivel de aprendizaje.

El funcionamiento de *CSS* se basa en la definición de estilos sobre selectores aplicados a los elementos que componen una página *HTML*. El lenguaje para definir los estilos está estandarizado por el *W3C*, lo cual permite encontrar rápidamente información y ayuda en páginas oficiales. Estos estilos se pueden establecer de tres formas:

a) Mediante texto en línea en el elemento *HTML*:
```
<body style="color: #fff">Hola mundo</body>
```

b) Estilo definido aparte, en el mismo documento *HTML* dentro de las etiquetas `<head></head>`:
```
<style>body{ color: #fff }</style>
```

c) Definición de estilo en un archivo `.css` aparte, enlazado en el documento *HTML*:
```
body{ color: blue; }
```

Aunque *CSS* tiene algunas limitaciones, como que los selectores no pueden aplicarse de manera ascendente (al contrario que con *XPath*), dificultad para el alineamiento vertical de elementos o ausencia de expresiones de cálculo numérico para especificar valores, sus ventajas compensan su utilización.

Entre ellas encontramos: centralización del estilizado del sitio web al completo en un único fichero separado de los documentos *HTML*; optimización del ancho de banda gracias a la definición de estilos aplicables a muchos elementos diferentes con mismos selectores; mejora en la accesibilidad del sitio web al abandonar antiguas malas prácticas como el uso de tablas, sin capacidad para redimensionarse según el tamaño del dispositivo sobre el que se visualiza, ahora pueden aplicarse diferentes estilos sobre los mismos elementos en circunstancias diferentes.

1.3.4 JavaScript

"*JavaScript* (abreviado comúnmente *JS*) es un lenguaje de programación interpretado, dialecto del estándar *ECMAScript*. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico" (Wikipedia: *JavaScript*)

Este lenguaje, ejecutado en el lado del cliente, nos permitirá construir interfaces interactivas con poco esfuerzo gracias a librerías como *jQuery*, con una API y documentación que hace el manejo de eventos, animaciones, selecciones y demás acciones sobre elementos *HTML* mucho más fácil. Esto lo consigue mediante la llamada funciones incluidas en la librería que unifican tareas arduas. Por ejemplo, vamos a ver la diferencia entre cambiar el fondo del elemento *body* de una página *HTML* con *JavaScript* puro y con la librería *jQuery*:

- **JavaScript**

```
function changeBackground(color) {
    document.body.style.background = color;
}
onload="changeBackground ('red');
```

- **jQuery**

```
$('body').css('background', '#ccc');
```

Como se puede apreciar, hemos reducido cuatro líneas a una realizando la misma acción, consiguiendo mayor legibilidad y velocidad de desarrollo.

1.3.5 *Asynchronous JavaScript And XML* (*AJAX*)

"*AJAX* es una técnica de desarrollo web para crear aplicaciones interactivas o *RIA* (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones." (Wikipedia: *AJAX*)

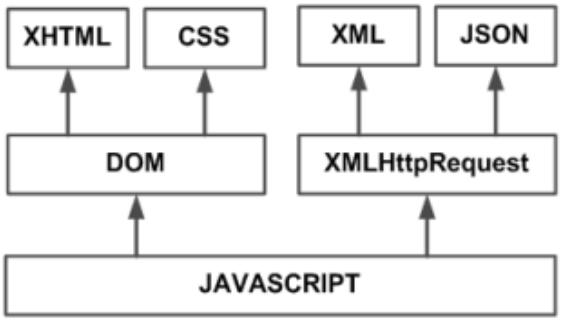


Imagen 4 Tecnologías agrupadas bajo el concepto de AJAX

Las tecnologías que forman *AJAX* son: *HTML* y *CSS* para la interfaz basada en estándares; *XML* y *JSON* para la transferencia de información; *Document Object Model* (D.O.M.[9]) para la manipulación de la interfaz dinámicamente; *XMLHttpRequest*, accedido a través de *jQuery* mediante el método `.ajax()`[10], que llevará a cabo la transferencia de la información de forma **asíncrona**; y por último, pero no menos importante, usaremos JavaScript para unir el resto de tecnologías [11].

Recordemos que *JavaScript* es un lenguaje de scripting del lado del cliente, por lo que necesitamos poder comunicarnos con el lado del servidor que estará procesando la lógica de la aplicación.

La posibilidad que ofrece enviar datos codificados en un estándar y recibirlos más tarde de forma asíncrona (sin recargar la página) permite una mayor usabilidad de la aplicación al no obligar al usuario a cambiar entre páginas continuamente.

De lo contrario, conllevaría a un aumento en el consumo de ancho de banda y la reducción de la capacidad de procesamiento de la computadora cliente, así como el incremento en el consumo de energía y la disminución de la vida de la batería en dispositivos móviles, resultando en una peor experiencia de usuario.

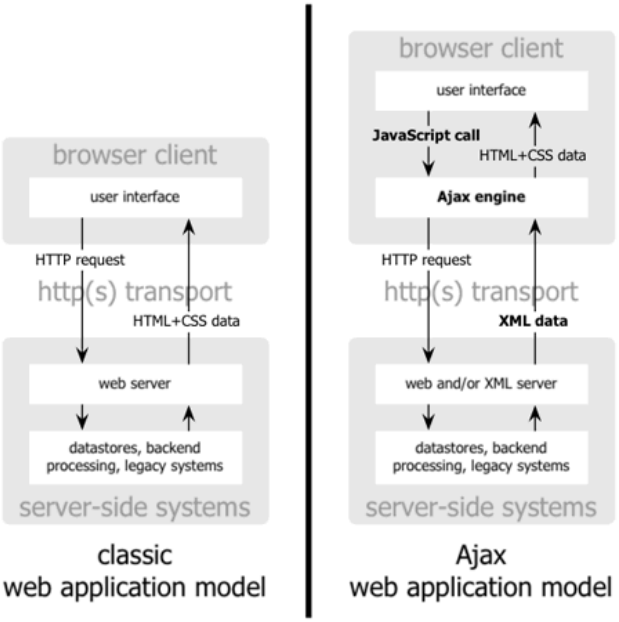


Imagen 5 Comparación entre modelos de aplicaciones web con y sin AJAX

1.3.6 Bootstrap

"Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web." **(http://getbootstrap.com/)**

Hasta ahora sabemos que necesitaremos un servidor web que soporte Apache, HTML para crear la interfaz, *CSS* para dotarle de estilo, *jQuery* para darle dinamismo, *PHP* para controlar la lógica de la aplicación y *AJAX* para comunicar la interfaz y la lógica. Sin embargo, la capa de presentación de nuestro programa ha de cumplir unos estándares si se quiere asegurar que su uso es compatible con los principales navegadores web y la amplia variedad de dispositivos con pantallas y formas de interactuación diferentes.

En este punto es donde aparece *Bootstrap*, un *framework* encargado de facilitar la tarea al diseñador web utilizando una serie de tecnologías y *APIs* ya conocidas que reducirán notablemente el tiempo de desarrollo de cualquier sitio web.

La documentación de *Bootstrap* ofrece y explica las características que el desarrollador podrá utilizar tras enlazar las librerías CSS y JS al documento HTML sobre el que se aplicará el entorno de trabajo: clases CSS predefinidas, elementos jerárquicos prediseñados (barras de navegación, tablas, botones, paneles, listas…), así como métodos *jQuery* para crear animaciones o manejar eventos.

El correcto uso de *Bootstrap* requiere seguir las guías de diseño de su página oficial, solo así nos aseguraremos de que nuestro sitio se adapte correctamente a

los distintos tamaños de pantalla y disponga de ayudas de accesibilidad para personas con capacidad sensorial reducida [14].

1.3.7 PHP

"*PHP* (acrónimo recursivo de *PHP*: *Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en *HTML*." (http://php.net/manual/es/intro-whatis.php: ¿Qué es *PHP*?)

Este lenguaje, de tipado débil, multiplataforma, con licencias de uso libre y de código abierto y de programación estructurada, trabaja en el lado del servidor, lo que significa que los programas escritos en *PHP* serán interpretados en tiempo de ejecución por el servidor web que los aloje. Esto es especialmente útil cuando estamos creando una lógica que debe funcionar en segundo plano y ser transparente al usuario.

PHP nos ayudará a crear la lógica de nuestro programa, hacer cálculos, guardar y recuperar información de la base de datos y, principalmente, establecer el flujo de trabajo de nuestra aplicación web.

El uso de este lenguaje puede incrementar la seguridad de nuestra aplicación al no permitir la visibilidad del código fuente de un archivo *PHP* desde un agente de usuario, ya que no es necesario su envío al cliente al establecer una comunicación *HTTP*. Esto nos puede ser especialmente útil si no queremos que el usuario pueda conocer cómo es construida la página que se le presenta. Además, solo podremos acceder al código de fuente de nuestro archivo desde el gestor de ficheros del sistema operativo host del servidor web. En el caso de tener que acceder remotamente, podemos usar el protocolo *FTP* para recuperar o subir ficheros al directorio del servidor en el que tengamos alojada la aplicación.

Pero además de seguridad y su fácil incrustado en *HTML*, *PHP* es un lenguaje de scripting (aunque en las últimas versiones ha sido adaptado para ofrecer programación orientada a objetos), originado principalmente para el ámbito de la web, lo que lo hace fácil de aprender para inexpertos. Es posible conseguir rápidamente resultados visibles con un par de líneas y una sencilla preconfiguración[12], al contrario, por ejemplo, que Java.

Aunque *Java* es un lenguaje precompilado (al contrario que *PHP* que es interpretado) y debería ser más rápido por mantener datos en memoria en lugar de inicializarlos en cada ejecución, la filosofía de *PHP* y su orientación al scripting permiten no usar librerías y clases redundantes que ocupan espacio en la memoria

virtual y disminuyen la capacidad de procesamiento y/o transferencia del servidor, lo que resulta, a fin de cuentas, en un menor tiempo de ejecución.

A pesar de la popularidad de Java y su experiencia en la programación orientada a objetos, la rapidez en el despliegue de un programa escrito en *PHP*, sumado a su corta curva de aprendizaje por su tipado débil y a las posibilidades que ofrece el scripting junto a su reciente adaptación a la programación orientada a objetos, lo convierte en el candidato ideal para un proyecto de dimensiones moderadas y fundamento educativo como el presente trabajo.

1.3.8 Base de datos

"Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto." (Wikipedia: Base de datos)

Bien, sabemos qué tecnologías podemos usar para crear nuestra interfaz, la lógica, conectar ambas y además usando entornos de desarrollo que nos facilitan y agilizan la labor. Sin embargo, estaríamos olvidando lo más importante: asegurar la persistencia de nuestros datos. Al fin y al cabo, la finalidad de nuestra aplicación es trabajar con ellos, guardarlos, recuperarlos, modificarlos…

En Internet podemos encontrar fácilmente numerosas soluciones a esta necesidad, pero, ¿cuál elegir? El tipo de base de datos más usado en la actualidad es el relacional (Edgar Frank Codd, 1970), en el que los datos son representados mediante tablas relacionadas entre sí. En la actualidad, es el modelo mayoritariamente elegido para resolver problemas reales y administrar datos dinámicamente (Wikipedia: Base de datos; Bases de datos relacionales).

Vamos, pues, a comparar los diferentes sistemas de gestión de base de datos relacionales y así elegir la opción más adecuada [15]:

	Creador	Fecha de la primera versión pública	Última versión estable	Licencia de software
Adaptive Server Anywhere	Sybase/iAnywhere	1992	10.0	Propietario
Adaptive Server Enterprise	Sybase Inc	1987	15.0	Propietario
ANTs Data Server	ANTs Software	1999	3.6	Propietario
DB2	IBM	1982	9	Propietario
Firebird	Firebird Foundation	25 de julio de 2000	2.5	Licencia Pública InterBase
Informix	Informix Software	1985	10.0	Propietario
HSQLDB	Hsqldb.Org	2001	1.9	Licencia BSD
Ingres	Berkeley University, Computer Associates	1980	2006	CA-TOSL
InterBase	Borland	1985	7.5.1	Propietario
SapDB	SAP AG	?	7.4	GPL con drivers LGPL
MaxDB	MySQL AB, SAP AG	?	7.7	GPL o propietario
Microsoft SQL Server	Microsoft	1989	2014	Propietario
MySQL	MySQL AB	Noviembre de 1996	5.0	GPL o propietario
Oracle	Oracle Corporation	1977	11g Release 2	Propietario
PostgreSQL	PostgreSQL Global Development Group	Junio de 1989	9.0	Licencia BSD
SmallSQL	SmallSQL	16 de abril de 2005	0.12	LGPL
SQLite	D. Richard Hipp	17 de agosto de 2000	3.6.16	Dominio público
Microsoft Access	Microsoft	1992	2013	Propietario
	Creador	Fecha de la primera versión pública	Última versión estable	Licencia de software

Imagen 5 Comparación de sistemas de gestión de bases de datos relacionales [16]

Dada la naturaleza del proyecto y su moderada complejidad, aprovecharemos soluciones libres y/o gratuitas que no aumenten el costo del trabajo. Según la tabla, nos centraremos en: *SapDB*, *MaxDB*, *MySQL*, *SmallSQL* y *SQLite*.

Desde la versión 7.5 en adelante, *SapDB* fue adquirido por *MySQL AB*, la misma compañía que desarrolla la base de datos de código abierto *MySQL*, siendo renombrada entonces a *MaxDB*, aunque el desarrollo todavía está a cargo de *SAP AG* (Wikipedia: *MaxDB*).

SmallSQL está escrito en *Java*, por lo que para su funcionamiento deberemos tener instalada y funcionando la máquina virtual de Java en el servidor de base de datos. Dado que vamos a usar *PHP* en lugar de *Java* como lenguaje de programación de la lógica de nuestra aplicación, no es viable darle una carga adicional al procesador al tener un programa adicional continuamente trabajando.

SQLite es una biblioteca escrita en *C* de pequeño tamaño que permite la gestión de bases de datos y es de dominio público. Es bastante popular en la actualidad, al igual que *MySQL*, por lo que vamos a hacer una comparativa entre ellos.

Según leemos en *digitalocean.com* [15] y *yises.com* [17], entre las ventajas de usar *SQLite* están:

- **Basado en fichero**: la base de datos al completa se almacena en un único archivo, lo que la hace extremadamente portable.

- **Basado en estándares**: aunque omite algunas características como `RIGHT OUTER JOIN` o `FOR EACH STATEMENT`.
- **Facilidad de uso durante el desarrollo y *testing*:** fácilmente escalable, con una amplia base de características de serie y con la simplicidad de trabajar con único fichero y una librería C enlazada.

Por el contrario, nos encontramos las siguientes desventajas:

- **No dispone de administración de usuarios:** lo cual supone que no existen perfiles con diferentes niveles de acceso.
- **Falta de optimización del rendimiento:** no optimiza consultas `SELECT`, ni almacena consultas en la *cache*.

En el otro lado, vamos a ver las ventajas de *MySQL*:

- **Fácil de trabajar**: sencilla instalación, disponibilidad de herramientas de terceros, incluyendo aquellas con representación visual, que facilita la toma de contacto con el desarrollo de la base de datos.
- **Repleto de funcionalidades**: soporta una amplia base de funcionalidad *SQL* esperada de un sistema de gestión de bases de datos.
- **Seguridad**: dada su popularidad, cuenta con numerosas características y parches de seguridad.
- **Escalable**: puede manejar grandes (e incrementales) cantidades de datos.
- **Veloz**: sus continuas revisiones y el uso de estándares le garantizan optimización en el código y operaciones eficientes.

MySQL también cuenta con limitaciones funcionales conocidas que podrían no ser adecuadas para proyectos muy complejo, ya que no asegura el cien por cien de fiabilidad dada la forma en que se trata algunas funcionalidades (manejo de referencias, transacciones o auditorías). Aun así, sus ventajas hacen que sea el candidato más adecuado para nuestro proyecto.

Con las tecnologías vistas hasta ahora, ya tenemos las herramientas necesarias para construir una aplicación web funcional:

- Diseño de la interfaz: *HTML* + *CSS* + *Bootstrap*.
- Animación de la interfaz: *Javascript* + *jQuery* + *AJAX*.
- Diseño de la lógica de negocio: *PHP*.
- Almacenamiento de datos: base de datos *MySQL*.
- Publicación de la aplicación: *HTTP* (servidor web).

A continuación, planificaremos el desarrollo del proyecto analizando primero los requisitos y diseñando la lógica que empleará.

2 Desarrollo

2.1 Análisis y diseño

Para implementar nuestra aplicación web primero necesitaremos recopilar toda la información posible de las partes interesadas, ya sea mediante entrevistas con cuestionarios o reuniones informales. Tras ello, analizaremos los datos obtenidos y definiremos una posible solución cuyo desarrollo tendrá un ciclo de vida.

La solución planteada se realizará abstrayendo la información recogida en entidades independientes y lógicas que interactuarán con el resto de componentes a través de un flujo de trabajo definido por un patrón de desarrollo de *software*.

2.1.1 Recopilación de información

La primera parte del análisis para la realización de nuestro proyecto consiste en la realización de entrevistas personales e informales con la parte interesada para realizar posteriormente la abstracción del problema, el modelado de nuestra solución y la arquitectura de *software* que emplearemos para implementar nuestra tarea.

A continuación, se incluyen muestras de resultados obtenidos en las reuniones y entrevistas, en las cuales preparamos un par encuestas con un conjunto de preguntas preparadas que nos ayuden a concretar los requisitos principales de nuestra aplicación.

Se obtuvieron los siguientes resultados:

P: ¿Quién está solicitando el desarrollo de este proyecto?

R: El Departamento de Sistemas Informáticos y de Computación (D.S.I.C.) de la Universidad Politécnica de Valencia.

P: ¿Qué objetivos tiene la aplicación y cómo se medirán?

R: Herramienta para la generación automática de grupos siguiendo distintos criterios.

P: ¿En qué ámbito será utilizada y para qué público objetivo?

R: Educación y empresa. El público objetivo será mayoritariamente profesores o jefes de proyecto y alumnos o trabajadores en el ámbito de la investigación

P: ¿Por qué se ha decidido crear un sitio web nuevo en lugar de adaptar el sitio web de la universidad?

R: Porque Poliformat no es adaptable y no cumple los requisitos que necesitamos.

P: ¿Qué os lo mejor y lo peor que tiene la web de la universidad actualmente?

R: Ventajas: registro de usuarios. Inconvenientes: poca flexibilidad.

P: ¿Qué tipos de usuarios utilizarán la web?

R: Principalmente organizador y participantes. También administradores.

P: La nueva aplicación, ¿será de acceso libre o requerirá invitación?

R: Para los participantes se requiere invitación expresa.

P: ¿Qué tipo de contenido será requerido principalmente?

R: Actividades, textos y formularios.

P: ¿Qué guías de diseño debería seguir la nueva aplicación (corporativo, minimalista…)?

R: Minimalista.

P: ¿Qué características técnicas debería incorporar?

R: Debería ser *responsive* (adaptable a la pantalla del dispositivo) y permitir el almacenamiento de los datos de los usuarios.

P: ¿Requerirá el nuevo sitio de mantenimiento periódico?

R: No.

P: ¿Existe algún tipo de información que requiera de modificación manual a través de una cuenta de administrador?

R: Sí: criterios de agrupación, información sobre los resultados e información sobre las actividades.

P: ¿Existe actualmente algún alojamiento en el que publicar la aplicación?

R: No.

P: Una vez acabado el desarrollo, ¿necesita promoción para darse a conocer?

R: Se hará difusión de la herramienta en foros especializados.

Posteriormente se especificaron los siguientes requisitos de acuerdo a las necesidades del interesado:

- El usuario tiene un panel principal en donde se ve un menú de opciones a la izquierda y en la parte central, todas sus actividades, que mientras no tenga ninguna estará vacío.
- El organizador invita a alumnos en sus actividades poniendo su lista de correos y les llega una invitación, con una contraseña.
  Un alumno aceptaría la invitación y podría modificar sus datos de acceso.
- Si el organizador va a invitar a otra actividad a un usuario que ya está en el sistema, podría salir un panel desplegable en el que sea fácil elegir.
- El alumno, al entrar vería sus actividades disponibles.
- Antes de participar en una actividad, los participantes deberían contestar los tests de autopercepción requeridos por ésta.
- Los tests de autopercepción son contestados una única vez, y sus resultados son persistentes.
- Los tests de autopercepción incluidos de momento serán *MBTI* y *Belbin*.
- Para realizar las agrupaciones, el organizador tendría un panel donde vería el listado de alumnos matriculados para la actividad, seleccionando los que entrarían en la agrupación (en principio todos), seleccionaría el tamaño de los grupos (mínimo y máximo) y un criterio de agrupación.
- Una vez realizadas las agrupaciones e iniciada la actividad, el organizador solo dispondrá de un botón de "Terminar actividad".
- En cuanto termine la actividad, el profesor tendrá un botón seleccionable de "activar evaluaciones", y a los alumnos se les activará el formulario de evaluación de su grupo.
- El alumno también podrá ver información referente al *feedback* de sus compañeros en las actividades en las que ha participado mediante gráficas.
- El profesor también podrá ver información sobre los resultados y las puntuaciones de las distintas evaluaciones de los grupos que han participado.

2.1.2 Arquitectura y proceso de *software*

Una vez obtenidos los datos, se pretende abstraer el problema para enfocar correctamente el desarrollo de una aplicación web.

Cuando analizamos las tecnologías a emplear en el ámbito de las aplicaciones web, decidimos elegir *PHP* por su facilidad a la hora de desarrollar y desplegar una aplicación, gracias a ser un lenguaje interpretado en lado del servidor, así como por su flexibilidad, potencia y alto rendimiento.

Gracias a la incorporación del soporte al paradigma de la programación orientada a objetos desde la tercera versión de PHP [18], es posible crear un código:

- **Reutilizable**: con clases definidas correctamente, es posible reusar código en otros áreas o proyectos.
- **Sostenible**: debido a la facilidad en la abstracción de los problemas resulta más fácil leer y comprender un programa diseñado orientado a objetos, ya que permite ocultar detalles de la implementación dejando visibles los más relevantes.
- **Modificable**: gracias a la especificación de las diferentes clases por separado, podemos añadir o eliminar funcionalidades con relativa sencillez.
- **Fiable:** al dividir el problema en partes más pequeñas es posible probar cada una por separado y así depurar código más rápidamente.

Ahora bien, para abstraer la información obtenida de las encuestas y entrevistas en clases interactivas y especificar un diseño lógico y coherente para nuestro *software*, debemos recurrir al lenguaje unificado de modelado (U.M.L. por sus siglas en inglés) que nos permite abstraer y visualizar cómodamente la definición de los objetos que emplearemos en nuestro proyecto (**Imagen 6**).

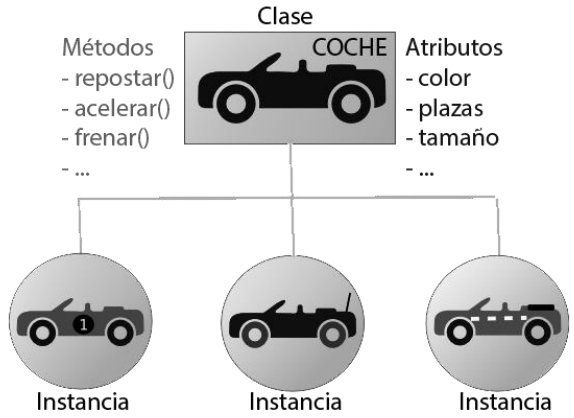


Imagen 6 Ejemplo del diseño lógico de una clase en programación orientada a objetos

Para representar nuestro problema, una aplicación web que administre y agrupe equipos de trabajo, definiremos mediante U.M.L. varios diagramas. Estos diagramas pueden desarrollarse mediante diferentes paquetes de programas, gratuitas o de pago. En nuestro caso emplearemos la versión gratuita de *Visual Paradigm*¸ válida para usos no comerciales [19].

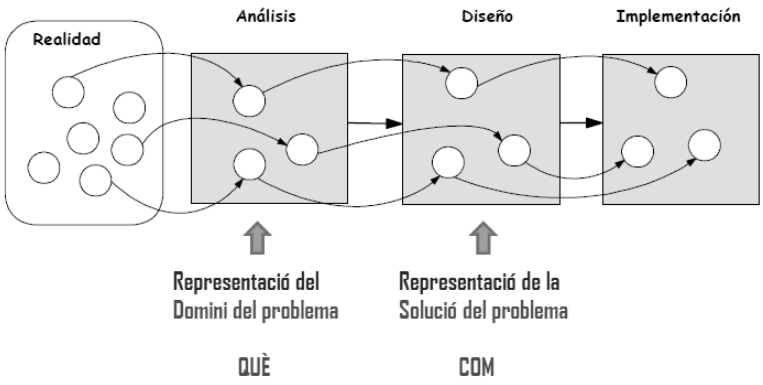


Imagen 7 Enfoque en el desarrollo orientado a objetos, descomposición del sistema en clases de objetos [20]

Pero antes de dar el salto al diseño de la lógica, vamos a repasar la arquitectura de *software* que empleará nuestra aplicación, esto es, la descripción de los diferentes subsistemas que la organizarán.

Para elegir el patrón arquitectónico adecuado se ha de tener en cuenta el tipo de sistema sobre el que va a funcionar el programa. En nuestro caso el tipo de sistema es **distribuido** (**Imagen 8**) por requerir de máquinas diferentes para procesar la información, en contraposición con los sistemas centralizados, cuya función es procesada en un único sistema informático dentro de un sistema más general.

En concreto, nuestro sistema se basa en el mismo que la arquitectura web: el modelo cliente/servidor (**Imagen 9**). Un navegador web (agente de usuario) sería el cliente y el servidor web donde está alojada la aplicación, el servidor.

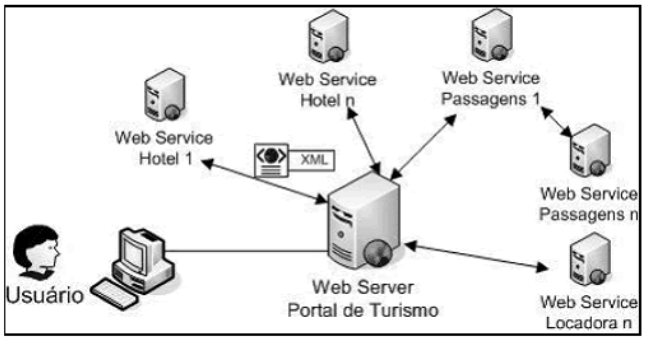


Imagen 8 Ejemplo de sistema distribuido: una página web

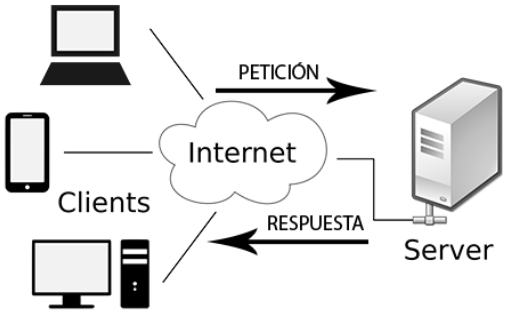


Imagen 9 Diagrama modelo cliente / servidor

Además, seguiremos el patrón arquitectónico de *software* llamado **modelo-vista-controlador** (M.V.C.) para separar la lógica y los datos de la interfaz de usuario, de forma que el mantenimiento y la ampliación de la aplicación por diferentes entidades sea menos costoso en tiempo. Además, permite comprender la lógica del programa mucho más fácilmente que simplemente leyendo el código fuente. (**Wikipedia:** Modelo–vista–controlador, **Imagen 10**).

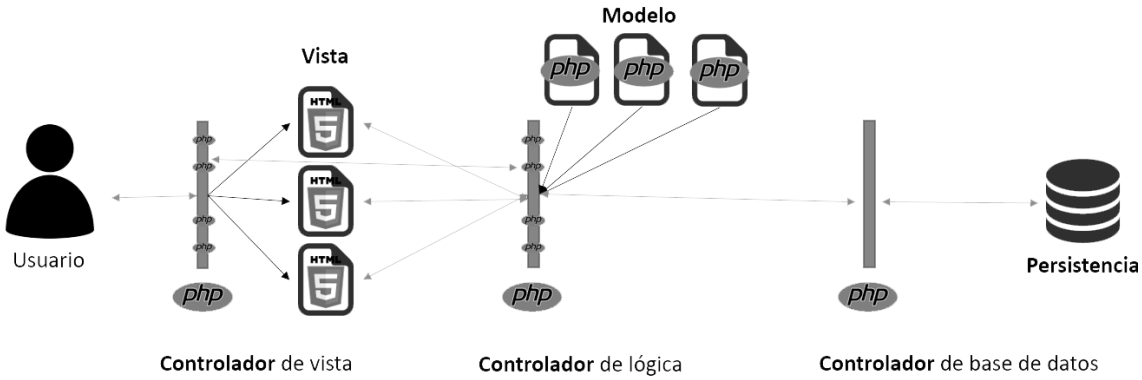


Imagen 10 Patrón Modelo-Vista-Controlador aplicado a una aplicación web

Para llevar a cabo el desarrollo de una aplicación web basada en la arquitectura cliente/servidor es necesario definir el proceso a llevar a cabo para conseguir un resultado exitoso. En el proceso de desarrollo de *software* se agrupan actividades genéricas que siempre aparecen relacionadas con la **especificación**, el **desarrollo**, la **validación** y la **evolución** del proyecto.

En la especificación se engloban el **análisis** y el **diseño**, en el desarrollo la **implementación**, en la validación las **pruebas** y la evolución hace referencia al **mantenimiento** de la aplicación en el paso del tiempo.

En la ingeniería del *software* existen varios modelos de ciclo de vida para este tipo de desarrollo según la envergadura del proyecto, la capacidad del equipo y otras circunstancias [21]. Estos modelos definen el proceso que se sigue desde que se empieza a analizar un problema hasta que se valida la solución aplicada. Algunos de estos modelos son:

- Codificar y corregir.
- Clásico o en cascada (**Imagen 11**).
    - Análisis de requisitos.
    - Diseño del Sistema.
    - Diseño del Programa.
    - Codificación.
    - Pruebas.
    - Verificación.
- Programación automática.

- Evolutivos:
    - Incremental.
    - En espiral.

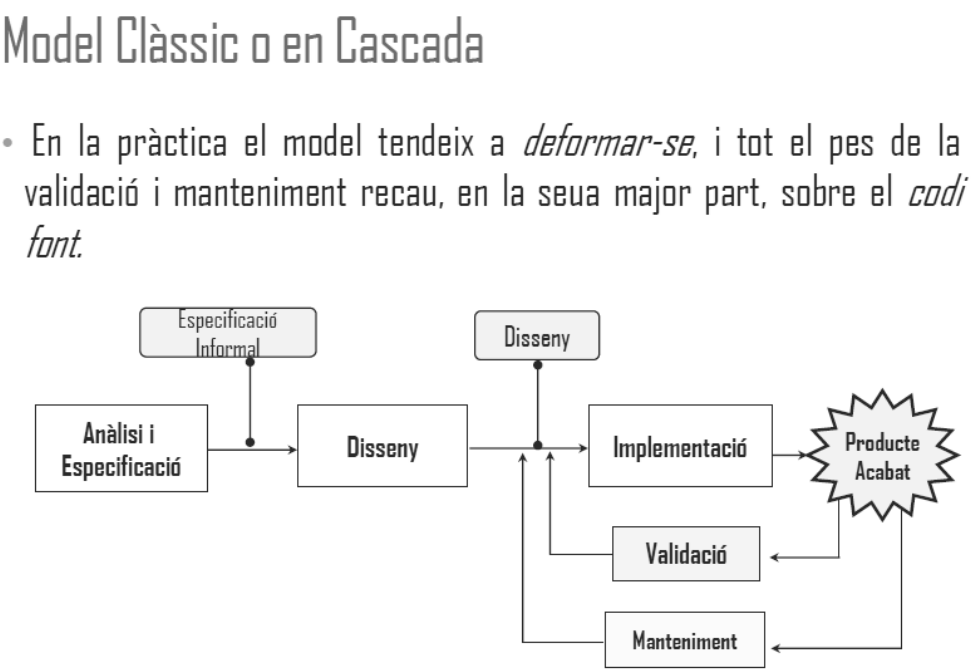


Imagen 11 Modelo de ciclo de vida de *software* [21]

En nuestro caso nos centraremos en el modelo clásico, al ser desarrollado por una única persona, tratarse de un proyecto con fines didácticos y seguir un modelo de desarrollo ya visto en asignaturas anteriores. Además, no necesitaremos prototipos completos de cada versión que requerirían de más tiempo para su realización. También aprovechamos las primeras fases de análisis y diseño de este modelo para optimizar los tiempos de desarrollo mediante la previsión de la lógica del sistema.

Con este no necesitamos mantener un histórico de diferentes versiones del producto, sino que realizaremos los pasos necesarios hasta llegar a un producto final, aunque éste al final también requiera mantenimiento.

Para conseguir llevarlo a cabo, analizaremos y modelaremos los requisitos de la aplicación extraídos de la fase de recopilación de información, abstrayendo esta información en objetos manipulables y con funcionalidad concreta (paradigma de la programación orientada a objetos).

2.1.3 Representación de la solución del problema: modelado

Clase	Atributos	Métodos	Tipo de dato (valor posible)
Actividad	Id	getId(), setId()	*Integer*
	Nombre	getName(), setName()	*String*
	Nombre corto	getShortName(), setShortName()	*String*
	Descripción	getDescription(), setDescription()	*String*
	Categoría	getCategory(), setCategory()	*String*
	Fecha de inicio	getStartDate(), setStartDate()	*Datetime*
	Fecha fin	getFinishDate(), setFinishDate()	*Datetime*
	Mostrar correos	getShowEmails(), setShowEmails()	*Integer* (0/1)
	Tamaño min. equipo	getMinTeamSize(), setMinTeamSize()	*Integer*
	Tamaño máx. equipo	getMaxTeamSize(), setMaxTeamSize()	*Integer*
	Visible	isVisible(), setIsVisible()	*Integer* (0/1)
	Id del organizador	getOrganizerId(), setOrganizerId()	*Integer*
	Iniciada	isStarted(), setStarted()	*Integer* (0/1)
	Finalizada	isFinished(), setFinished()	*Integer* (0/1)
	Equipos formados	teamsDone(), setTeamsDone()	*Integer* (0/1)
	Encuestas enviadas	surveySent(), setSurveysSent()	*Integer* (0/1)
	Equipos	getTeams(), setTeams()	*Array [*Equipo*]*
Equipo	Id	getId(), setId()	*Integer*
	Tamaño equipo	getTeamSize(), setTeamSize()	*Integer*
	Participantes	getParticipants(), setParticipants()	*Array[*Participante*]*
Test	Id	getId(), setId()	*Integer*
	Título	getTitle(), setTitle()	*String*
	Descripción	getDescription(), setDescription()	*String*
	Preguntas	getQuestions(), setQuestions()	*Array[*Pregunta*]*
Encuesta	Id	getId(), setId()	*Integer*
	Título	getTitle(), setTitle()	*String*
	Descripción	getDescription(), setDescription()	*String*
	Preguntas	getQuestions(), setQuestions()	*Array[*Pregunta*]*
Pregunta	Id	getId(), setId()	*Integer*

	Descripción	getDescription(), setDescription()	*String*
	Subpreguntas	getSubquestions(), setSubquestions()	*Array[*Subpregunta*]*
Subpregunta	Id	getId(), setId()	*Integer*
	Descripción	getDescription(), setDescription()	*String*
	Valor	getValue(), setValue()	*String*

La clase principal definida es **Actividad**, la cual agrupa todos los atributos requeridos para administrar las tareas a las que se unirán los **Participantes**. Al crear una actividad, se seleccionan los **Tests** que serán requeridos a los participantes antes de unirse a esta. Los resultados de los tests serán utilizados por el algoritmo formador de equipos proporcionado por el interesado en el proyecto, una vez el **Organizador** active la generación de los **Equipos** en la actividad.

Tras formar los conjuntos, la actividad cambiaría a un estado *En curso* hasta alcanzarse la fecha límite o hasta que el organizador que la administra decidiera finalizarla manualmente. Al ocurrir esto, el organizador dispone de un botón para solicitar a los participantes de la actividad que rellenen una **Encuesta** de valoración, cuyos resultados se usarán para generar gráficas dinámicas que recogen datos estadísticos sobre la participación en las actividades, los distintos perfiles de personalidad o la satisfacción de los integrantes, entre otros.

Los tests y encuestas están formados por **Preguntas** y éstas, a su vez, de **Subpreguntas**. Las últimas incluyen distintos apartados dentro de una misma pregunta y los valores obtenidos. Por ejemplo, preguntas con varios subapartados y diferentes opciones, como seleccionar diferentes atributos para diferentes **Usuarios** bajo un único enunciado.

Los **Administradores**, por su parte, serían capaces de crear, modificar y eliminar contenido, es decir: actividades, usuarios, equipos e incluso resultados de encuestas o tests.

Así pues, una vez analizada la estructura que sustentará nuestra *web app*, las diferentes clases que definirán nuestro modelo de negocio, es el momento de llevar la teoría a un gráfico en el que nos sea más fácil y rápido su entendimiento. El resultado, visto en lenguaje unificado de marcas (U.M.L.) y diseñado mediante *Visual Paradigm Community Edition 13.0*, aparece en la **Imagen 12**.

Usuario	Id	getId(), setId()	*Integer*
	E-mail	getEmail(), setEmail()	*String*
	Contraseña	getPassword(), setPassword()	*String*
	Nombre	getName(), setName()	*String*
	Apellidos	getSurnames(), setSurnames()	*String*
	N.I.F.	getDni(), setDni()	*String*
	Ciudad	getCity(), setCity()	*String*
	País	getCountry(), setCountry()	*String*
	Imagen	getImage(), setImage()	*Object*
Participante	Extiende a Usuario	Extiende a Usuario isParticipant()	*Integer* (1)
Organizador	Extiende a Usuario	Extiende a Usuario isOrganizer()	*Integer* (1)
Administrador	Extiende a Usuario	Extiende a Usuario isAdmin()	*Integer* (1)

A continuación, emplearemos el lenguaje unificado de modelado para organizar y especificar las relaciones entre los objetos (**Imagen 13**, **Imagen 14**, **Imagen 15**, **Imagen 16** e **Imagen 17**).

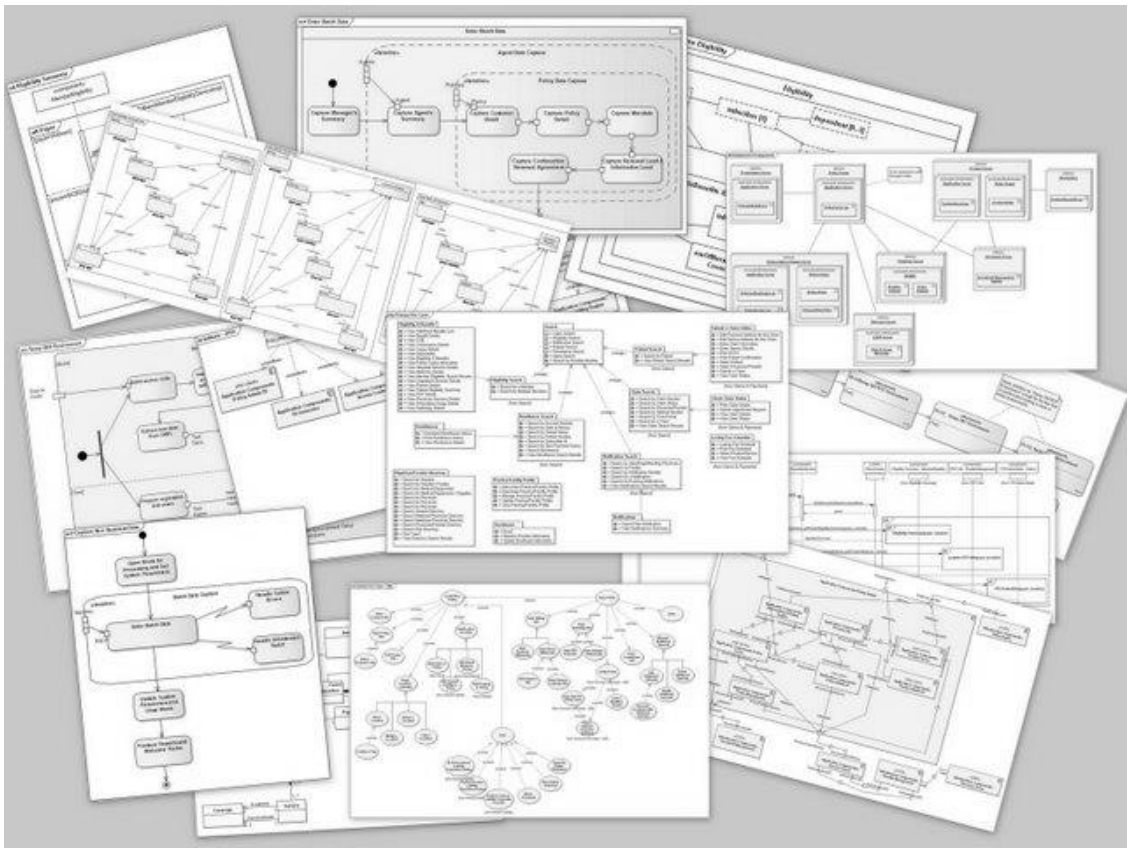


Imagen 12 Collage de diagramas UML (**Wikipedia**: Lenguaje unificado de modelado)

a) **Diagramas de clases** para concretar los objetos y definir sus atributos y métodos. Primero especificaremos manualmente los datos en la siguiente tabla.

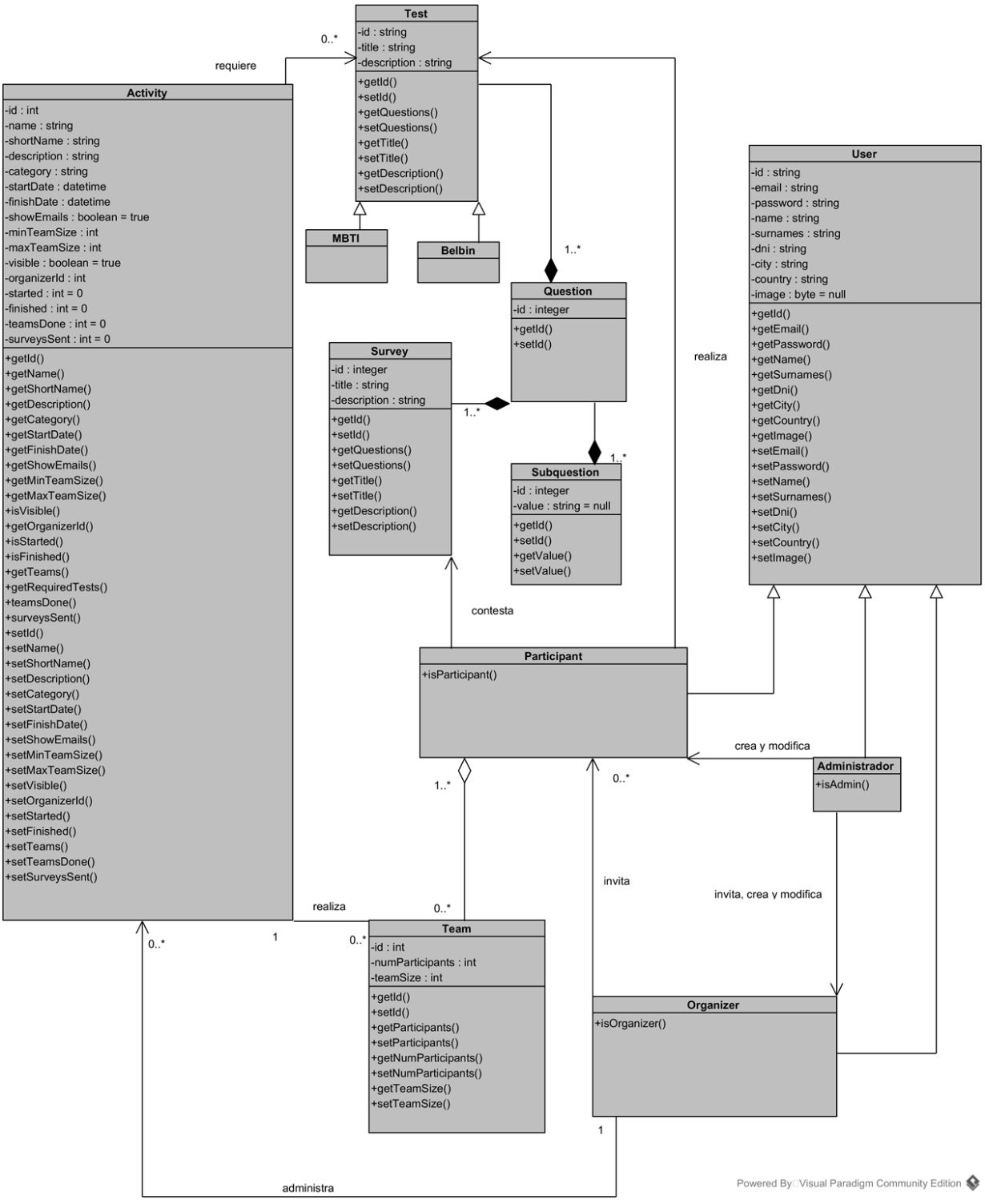


Imagen 13 Diagrama de clases de la aplicación web en desarrollo

Pero además de diseñar las clases de la lógica, es necesario crear otro diagrama para especificar las tablas que definirán la estructura de la base de datos que permitirá la persistencia de nuestros datos (**Imagen 14**). Este nuevo diagrama de clases será muy parecido al anterior, ya que está relacionado directamente. Los objetos creados a partir de las clases de la lógica deben ser capaces de insertar y recuperar información de dicha base de datos, por lo que las estructuras sobre las que se almacenan han de ser coherentes.

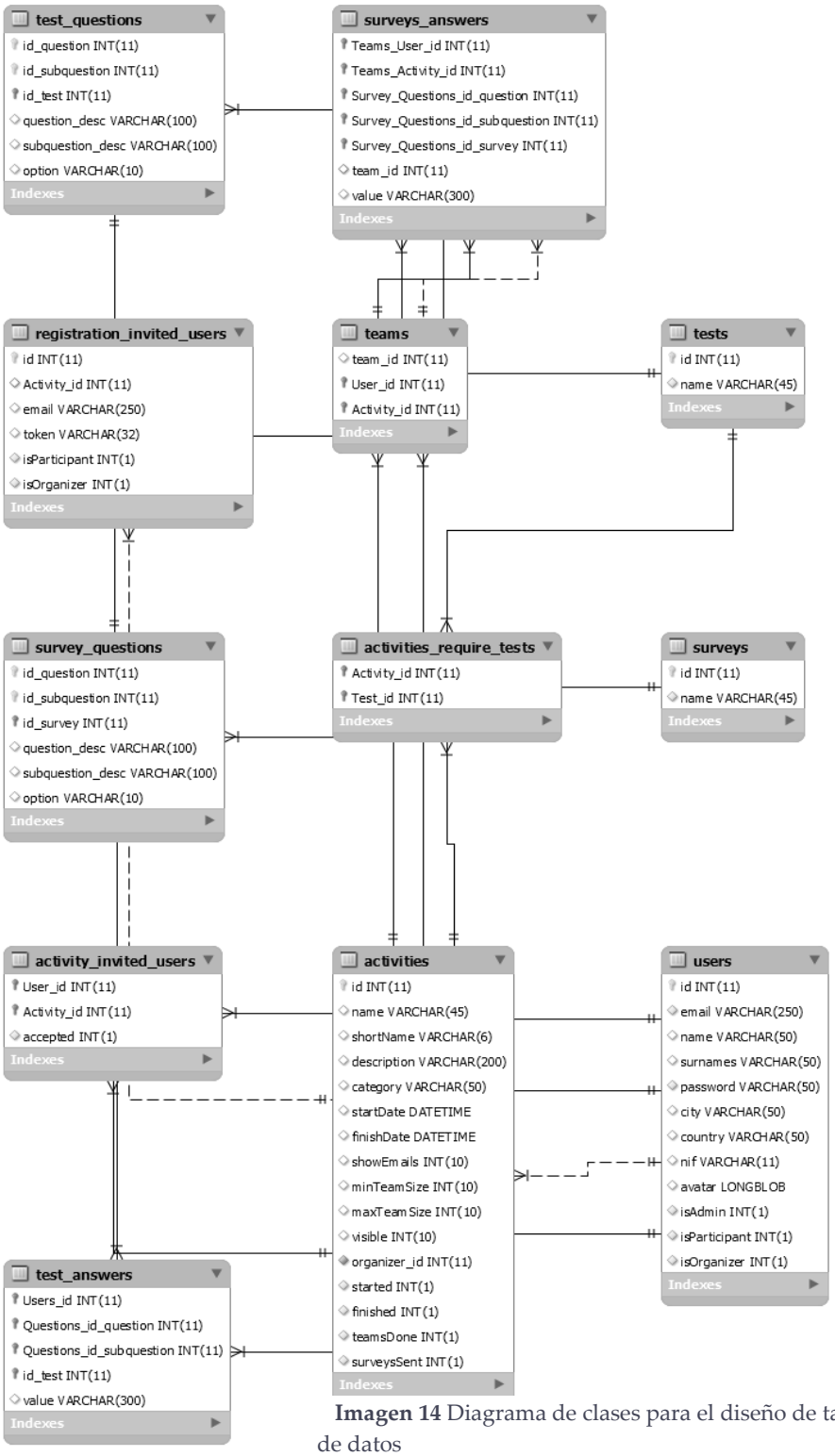


Imagen 14 Diagrama de clases para el diseño de tablas de la base de datos

b) **Diagramas de casos de uso** para especificar los actores que intervendrán en la aplicación y las funciones que realizará el sistema.

Los resultados en el análisis de clases ya nos muestran la necesidad de categorizar diferentes tipos de usuarios, los usuarios que visitarán nuestra web y que dispondrán de permisos concretos para acceder a determinados contenidos o funciones.

De ello deducimos la participación en el sistema de tres actores diferentes: participantes, organizadores y administradores.

Las acciones básicas que podrá realizar un **participante** son:

- Registrarse en el sistema.
- Listar las actividades disponibles.
- Aceptar la invitación a una actividad.
- Rellenar un test.
- Rellenar una encuesta.
- Mostrar resultados de encuestas en gráficas.
- Ver y modificar los datos de la cuenta registrada.

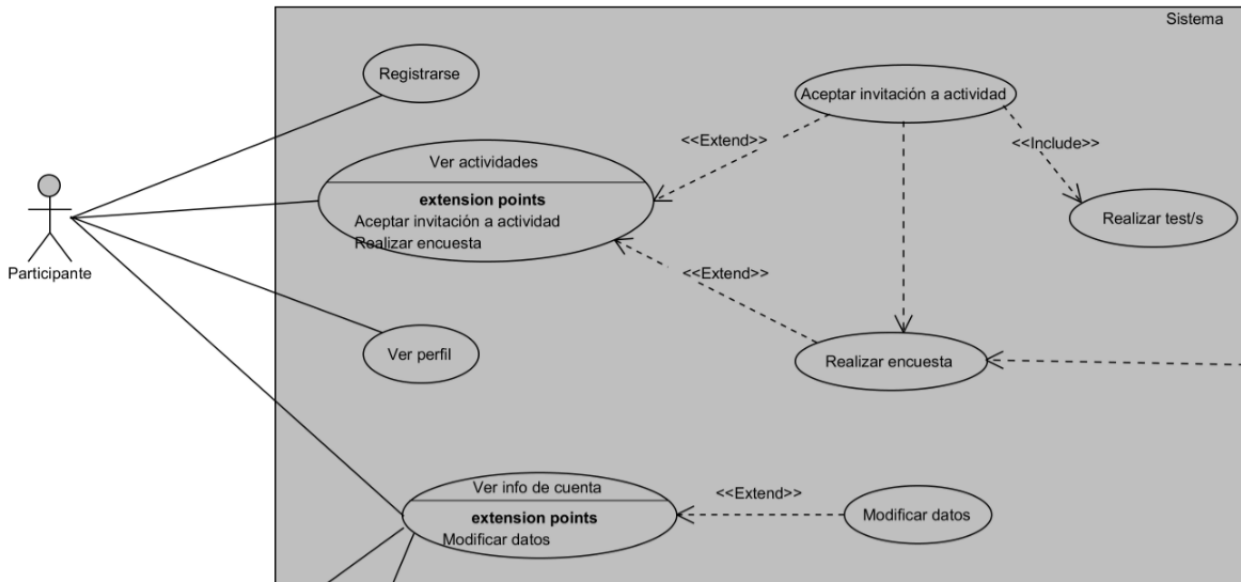


Imagen 15 Diagrama de casos de uso de un **Participante**

A continuación, veremos el caso para organizadores y administradores.

Un **organizador** comparte con el participante algunas acciones, como el registro o la visualización y modificación de los datos de su cuenta. Sin embargo, cuenta con más acciones relacionadas con su privilegio como organizador.

- Crear y modificar actividades.
- Realizar las agrupaciones de equipos en una actividad.
- Iniciar y terminar actividades.
- Invitar a participantes a una actividad.
- Mostrar resultados en forma de gráficas de las encuestas completadas de actividades finalizadas.

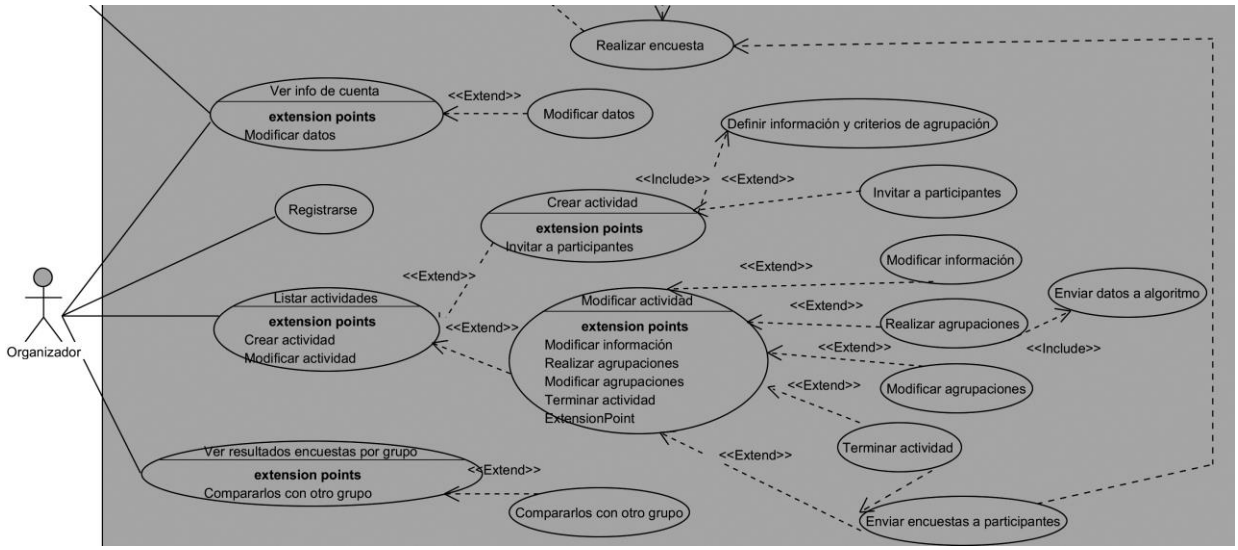


Imagen 16 Diagrama de casos de uso de un **Organizador**

En el caso de un **administrador**, tal y como analizamos en el diagrama de clases, serían capaces de crear, modificar y eliminar usuarios, tests y encuestas. El diagrama de casos de uso para este rol quedaría como sigue:

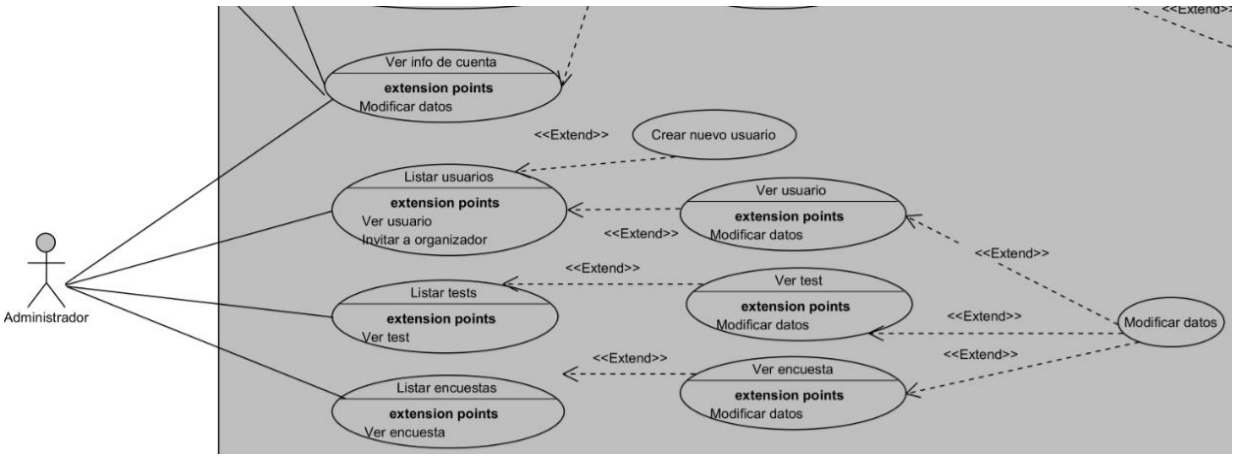


Imagen 17 Diagrama de casos de uso de un **Administrador**

2.1.4 Diseño de la interfaz

Teniendo de guía las especificaciones de los diagramas de clases y casos de uso, se han realizado bocetos de las diferentes páginas que compondrán nuestra aplicación web. El diseño de estos *mockups* agiliza el tiempo de desarrollo al poder percatarse de posibles errores visuales u organizativos que pudieran darse en el flujo y presentación del programa.

Es importante tener en cuenta el público objetivo que utilizará nuestro programa: mayoritariamente joven, acostumbrado a Internet y a dispositivos móviles, estudiantes, equipos de trabajo… Dado el amplio alcance y la posibilidad de que sea usada por personas con posibles limitaciones sensoriales, hemos de tener en cuenta guías de usabilidad y accesibilidad que garanticen la satisfacción del usuario al usar nuestro producto.

Por ello, seguiremos estándares web que permitan la accesibilidad desde múltiples navegadores, tamaños de pantalla o dispositivos de visualización (pantalla o síntesis de voz, por ejemplo); mejoren la usabilidad y el reconocimiento de formas y patrones mediante colores con contraste, estructura bien definida y otras técnicas.

En concreto, se dice que un producto es usable cuando [22]:

- Puede ser utilizado en forma adecuada, eficiente y satisfactoria por la mayoría de los posibles usuarios.
- Es tan fácil de aprender a usar que no requiere manuales.
- Puede ser utilizado por personas con diferentes habilidades o discapacidades.
- Cualquier persona, sin importar su edad o cultura, puede usarlo.
- Evite que el usuario cometa errores.

Además de tener en cuenta la accesibilidad y la usabilidad, necesitamos diseñar un logo y un nombre de acuerdo a la temática dinámica, activa y educativa del proyecto:

Imagen 18 Logo y nombre elegidos para la aplicación

Para empezar con el diseño de la interfaz, haremos un análisis mediante las preguntas "¿Qué?", "¿Quién?", "¿Cuándo?", "¿Cómo?" y "¿Dónde?" de los tipos de contenido que hemos representado para determinar así el conjunto de páginas que formarán la aplicación.

Qué

Pregunta	**Respuesta**
Qué es *NiceTeam*	Página explicativa "Acerca de"
Qué puedo hacer aquí	

Quién

Pregunta	**Respuesta**
Quién puede acceder al sitio web	Participantes, Organizadores, Administradores
Quién invita a usuarios a registrarse como Participantes	Organizadores
Quién invita a usuarios a registrarse como Organizadores	Administradores
Quién crea o modifica una actividad	Organizadores
Quién se une a una actividad	Participantes
Quién realiza los tests de autopercepción	Participantes
Quién realiza las encuestas de satisfacción	Participantes
Quién crea tests de autopercepción	Administradores
Quién crea o modifica la encuesta de satisfacción	Administradores
Quién ve los resultados de las encuestas sobre Actividades y Grupos	Organizadores
Quién ve los resultados personales de los Participantes	Cada Participante, los suyos propios
Quién ve, crea o modifica cualquier usuario del sitio	Administradores
Quién ve todas las actividades registradas en el sistema	Administradores

Cuándo

Pregunta	Respuesta
Cuándo se establecen la información y requisitos de una Actividad	Al crear la Actividad
Cuándo invita un Organizador a unirse a Participantes	Al crear o modificar la actividad
Cuándo se forman los equipos de una Actividad	Al modificar la actividad
Cuándo puede un Participante unirse a una Actividad	Cuando un Organizador le invite
Cuándo realiza un Participante los tests de autopercepción requeridos para una Actividad	Antes de unirse a la Actividad
Cuándo activa un Organizador la realización de encuestas	Al terminar la Actividad
Cuándo realiza un Participante la encuesta de satisfacción	Al terminar la Actividad y cuando el Organizador active el envío de las encuestas
Cuándo pueden verse los resultados de las encuestas	Desde que se activa el envío de encuestas y al menos un Participante de la actividad la haya realizado

Cómo

Pregunta	Respuesta
Cómo se registra en el sitio web	Mediante invitación
Cómo se accede al sitio web	Mediante usuario y contraseña
Cómo se crea una Actividad	Desde la página de Actividades Nuevas
Cómo se une a una Actividad	Desde la página de Actividades Nuevas
Cómo se forman los equipos	Desde la página de modificación de la Actividad (Actividades Nuevas)
Cómo abandona un Participante una Actividad	El Organizador ha de expulsarlo desde la página de modificación de la Actividad
Cómo modifica un usuario los datos de su cuenta	Desde la página de Mi cuenta
Cómo se activa el envío de encuestas a los Participantes	Desde la página de Actividades Finalizadas
Cómo se realiza la encuesta de satisfacción	Desde la página Actividades Finalizadas
Cómo navego a través del sitio web	Mediante dos menús:

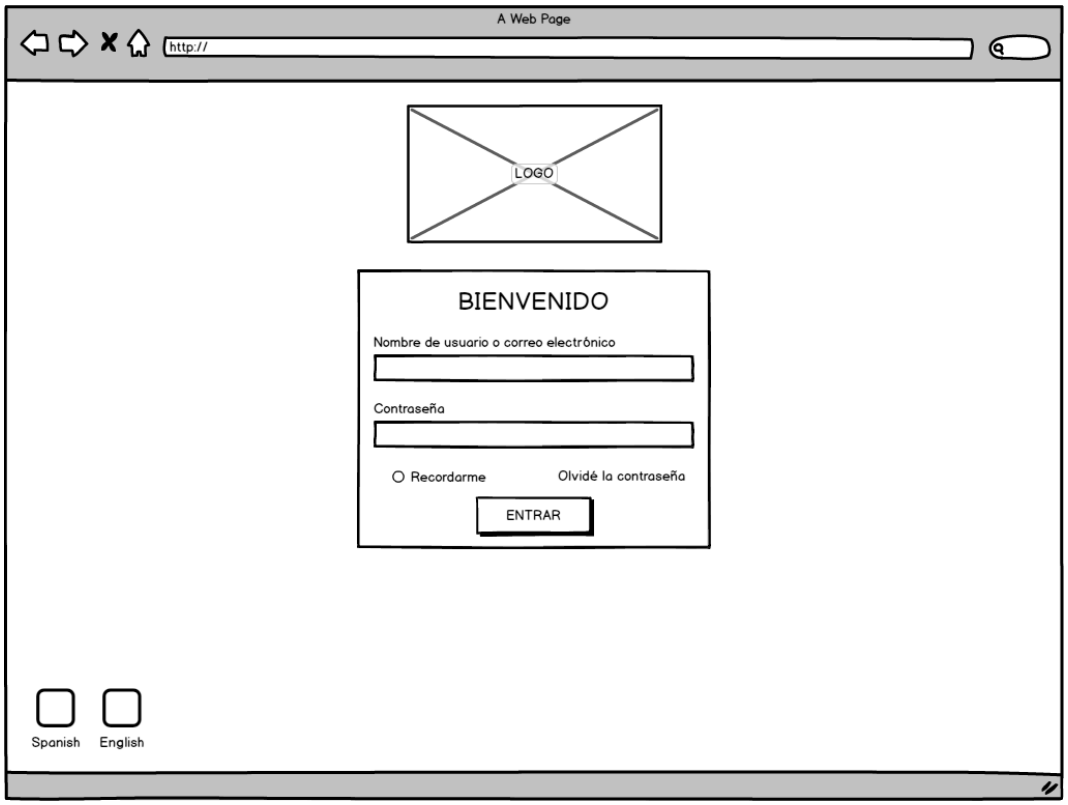
	- uno horizontal superior con opciones de cuenta e información del sitio; - otro vertical lateral con las áreas de acceso del tipo de usuario correspondiente.

Dónde

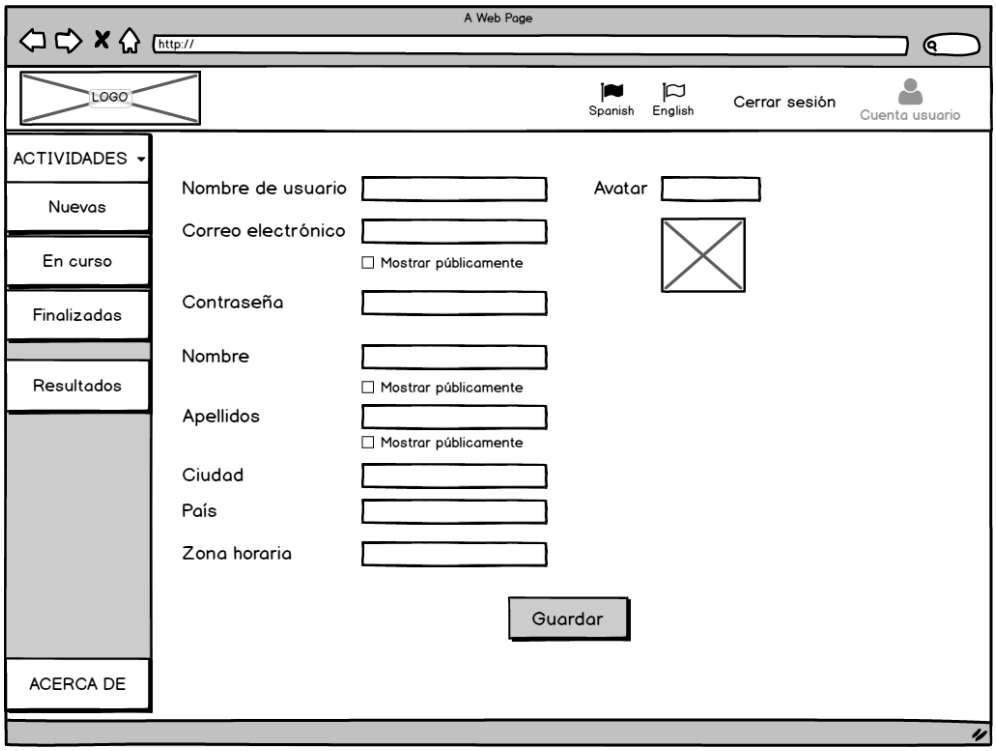
Pregunta	Respuesta
Dado que es una aplicación web accesible desde cualquier punto de Internet y no ofrece ningún producto, evento o servicio físico, no es necesario preguntarse “Dónde” se realiza la acción.	

De estos resultados, realizamos varios bocetos que nos ayudarán a detectar posibles errores o incoherencias y a validar nuestros diseños antes de implementarlos. En concreto aquí se presentan una muestra de ellos:

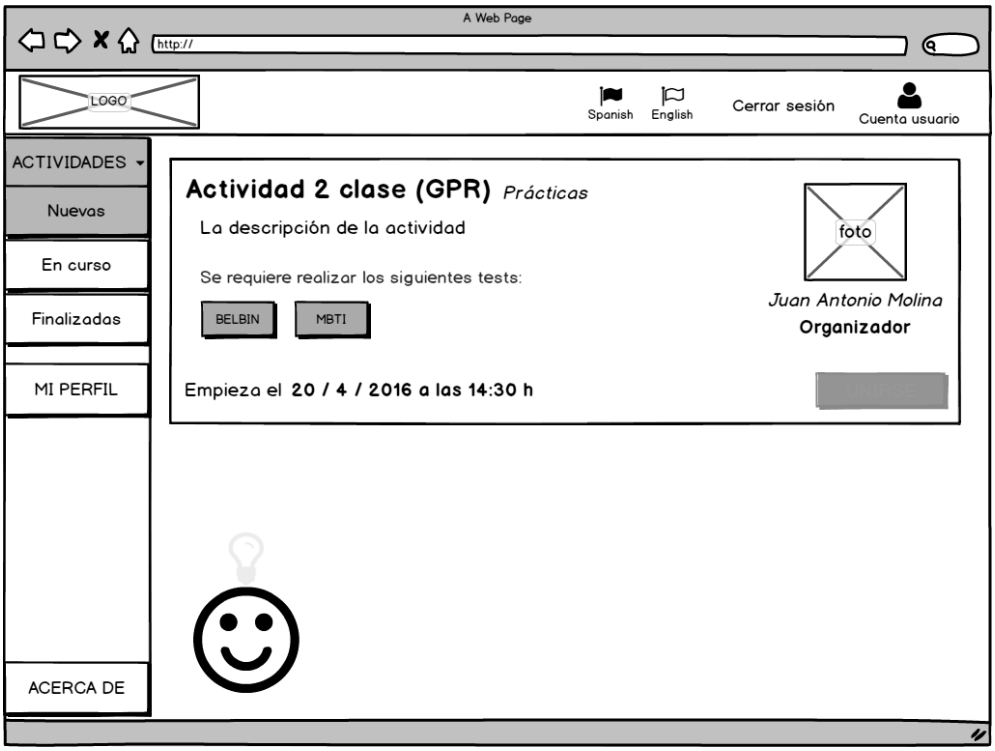
Página de conexión a la aplicación



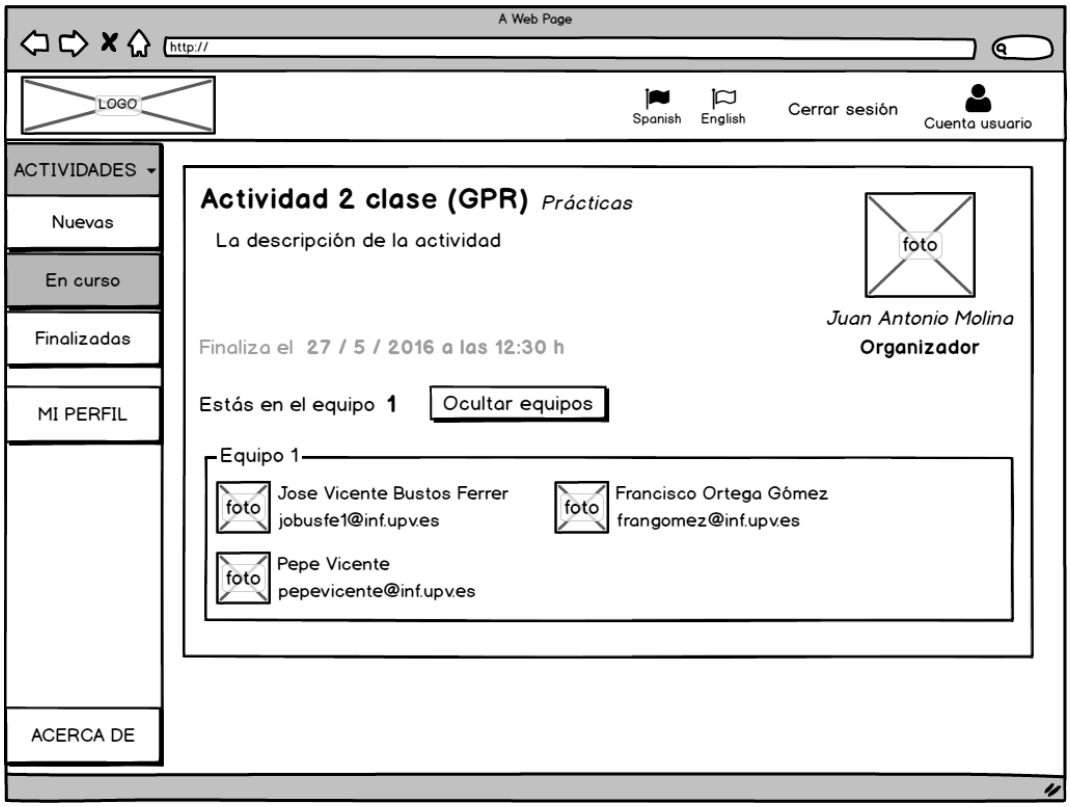
Página "mi cuenta"



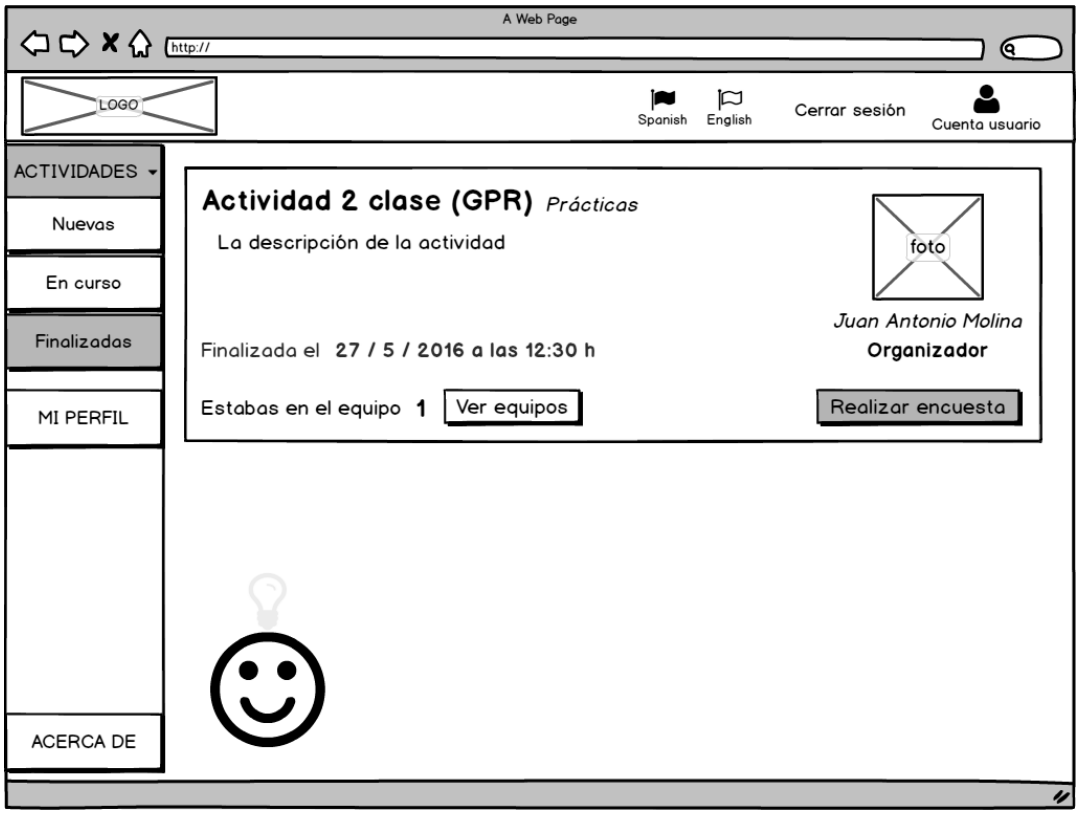
Actividades nuevas pendientes de aceptar para un participante



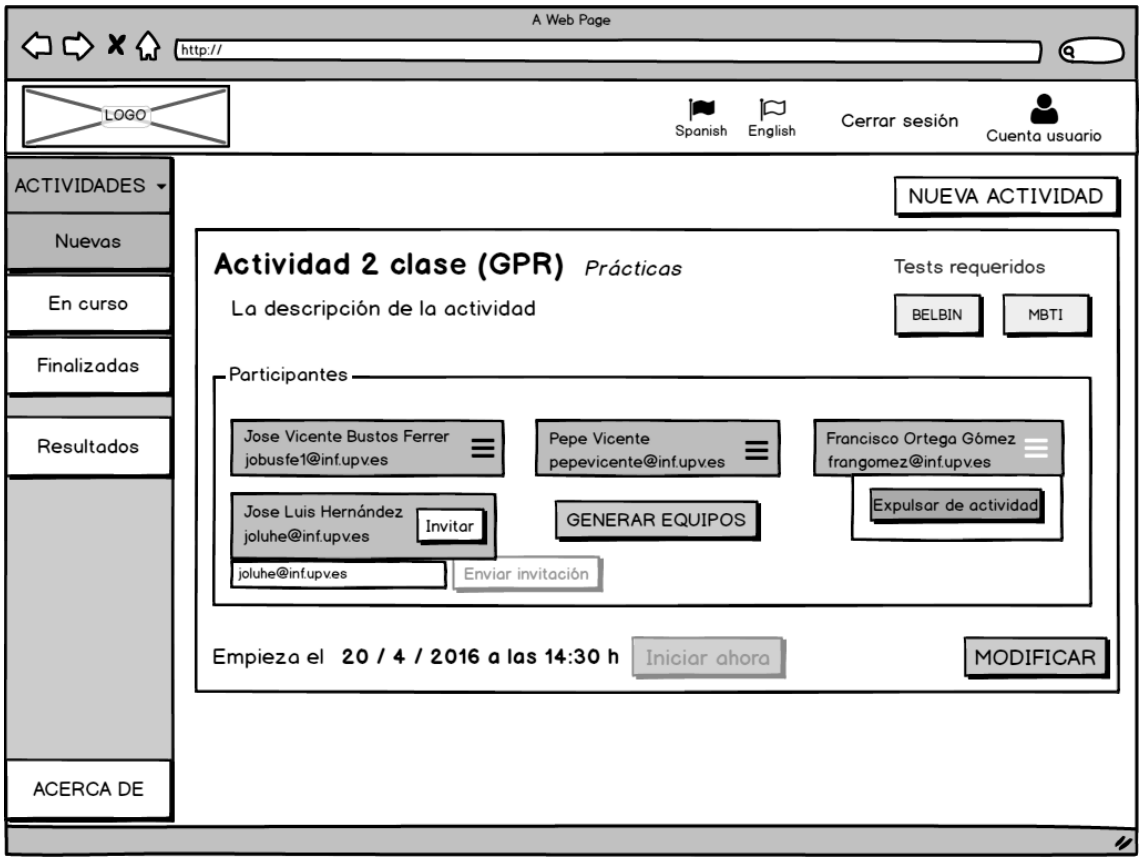
Actividades actualmente en curso, vistas por un participante



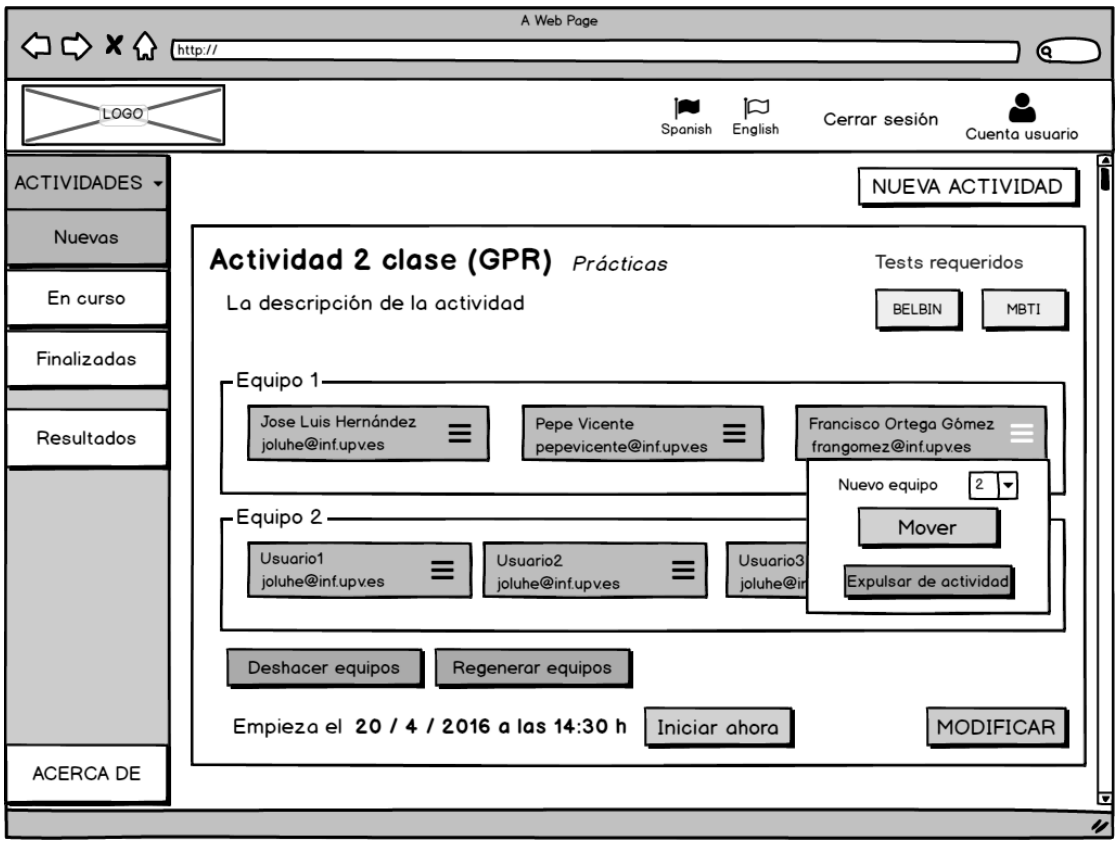
Actividades realizadas por un participante, ya finalizadas



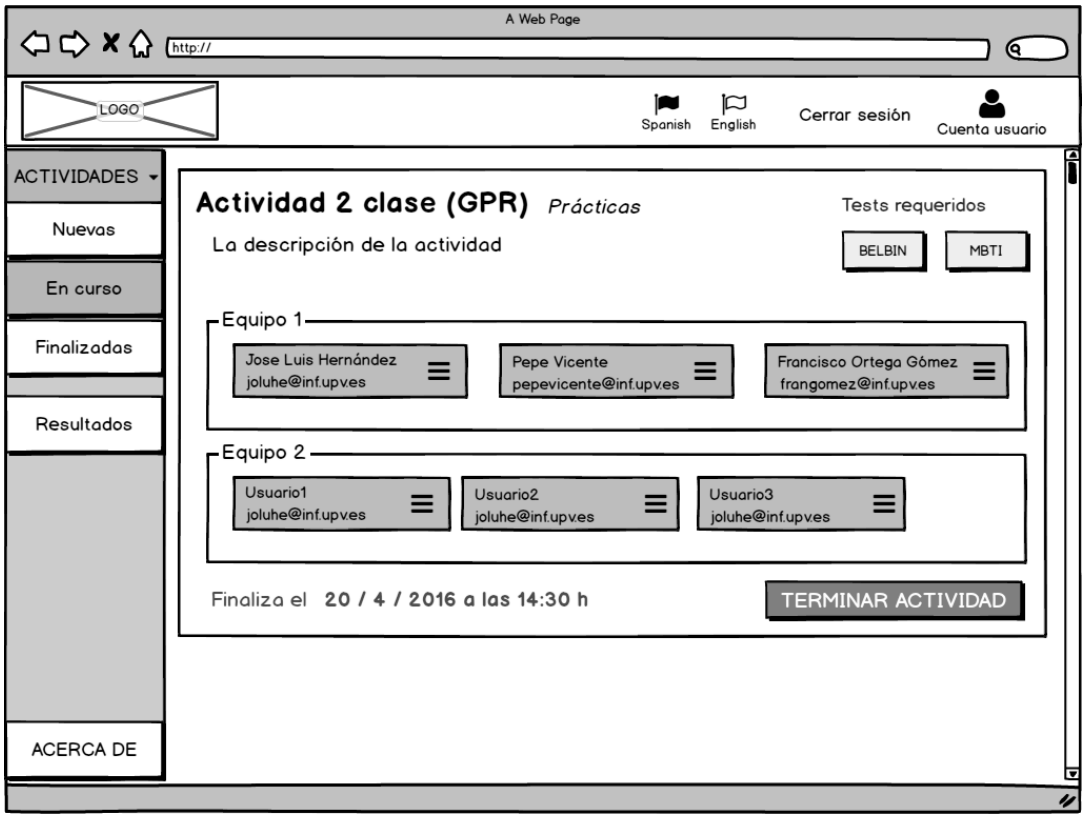
Actividad aún no iniciada por un organizador, sin equipos generados



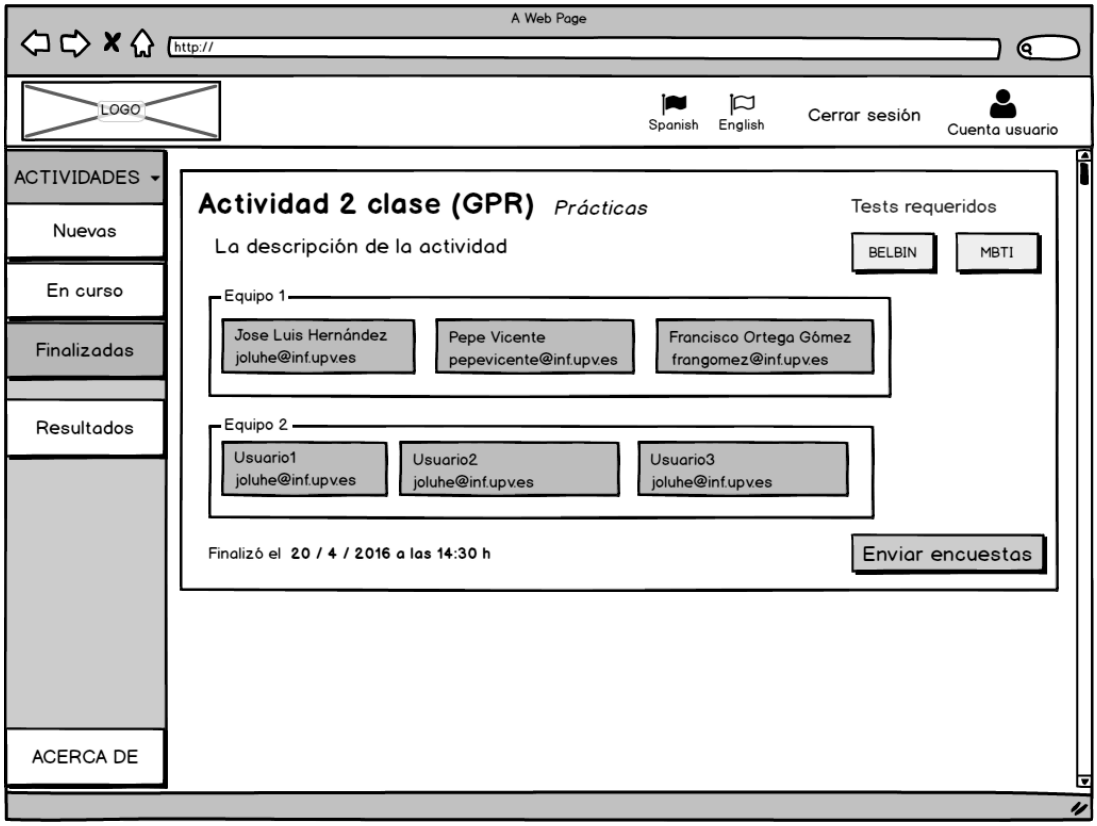
Actividad aún no iniciada por un organizador, con los equipos formados



Actividad en curso, con posibilidad de terminarla manualmente



Actividad finalizada, ahora es posible enviar las encuestas a los participantes



Estos bocetos han sido diseñados mediante una versión de prueba de *Balsamiq Mockups* [23].

2.2 Entorno de desarrollo y librerías

Teniendo claro lo que queremos conseguir y conociendo la organización y funcionamiento del sistema que vamos a implementar, es hora de elegir el entorno de desarrollo sobre el que vamos a programarlo y probarlo. Además, elegiremos e instalaremos los servidores web y de base de datos que necesitaremos para ello.

Tras usar un buscador online es posible encontrar rápidamente distintas comparativas sobre las que conocer entornos de desarrollo integrados y elegir el que más nos convenga [24].

De entre las opciones más populares para programar aplicaciones en PHP con soporte para HTML, CSS y Javascript podemos encontrar *software* como:

- **Netbeans**: entorno de desarrollo integrado libre, programado principalmente en Java con capacidad de extensión de características mediante módulos (**Imagen 19**). Además, es un producto libre, gratuito y sin restricciones de uso (**Wikipedia:** NetBeans). Cuenta con paquetes de instalación preparados para soportar el lenguaje y entorno de programación deseados. Por ejemplo, podemos descargar el instalador de Netbeans preparado para desarrollar en PHP, HTML, Javascript y configurar rápidamente un entorno de despliegue para el testeo de la aplicación.

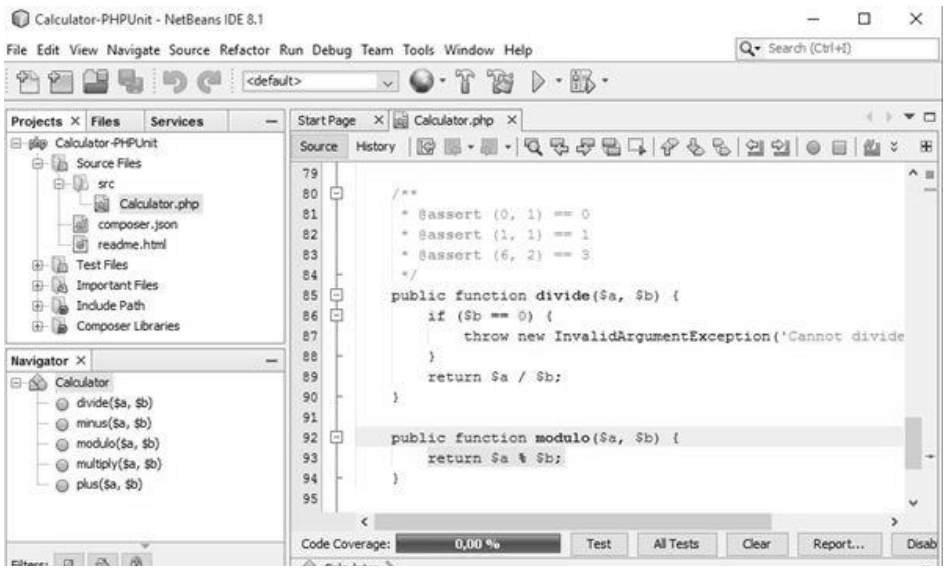


Imagen 19 Captura del entorno de desarrollo Netbeans ejecutándose en Windows 10

- **Eclipse:** popular entorno de desarrollo con una gran comunidad detrás (**Imagen 20**). Escrito también en Java, actualmente está licenciado mediante *Eclipse Public License*, reconocida como licencia de software libre por la *Free Software Foundation*, pero incompatible con la Licencia pública general de GNU (*GNU GPL*). A pesar de su popularidad, hay cierta controversia respecto a su consumo de recursos del sistema, lo que lo haría un entorno de desarrollo lento en computadores poco potentes [25].

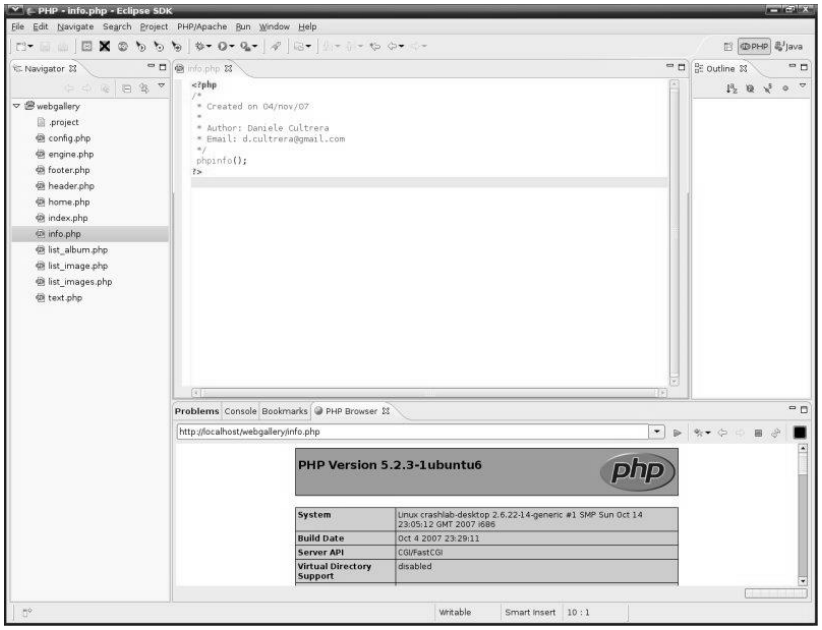


Imagen 20

Captura de Eclipse ejecutándose en Ubuntu

- **Aptana:** Aptana Studio (**Imagen 21**) es un entorno de desarrollo integrado de software libre basado en Eclipse y desarrollado por Aptana Incorporated. Provee soporte para lenguajes como: Php, Python, Ruby, CSS, AJAX, HTML y Adobe AIR. Tiene la posibilidad de incluir complementos para nuevos lenguajes y funcionalidades (**Wikipedia:** Aptana Studio).

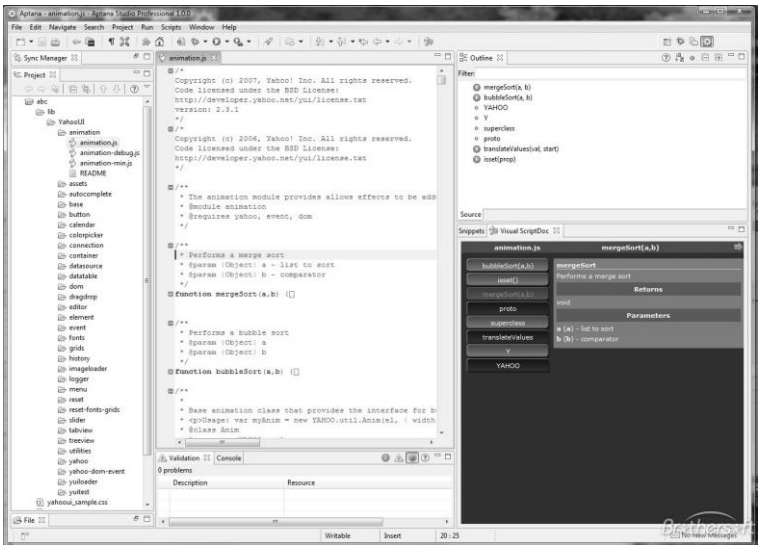


Imagen 21

Captura de Aptana Studio

- **Sublime Text:** editor de texto y editor de código fuente está escrito en C++ y Python para los *plugins* (**Imagen 22**). Puede descargarse y evaluarse de forma gratuita. Sin embargo, no es *software* libre y se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene fecha de caducidad (**Wikipedia**: Sublime Text).

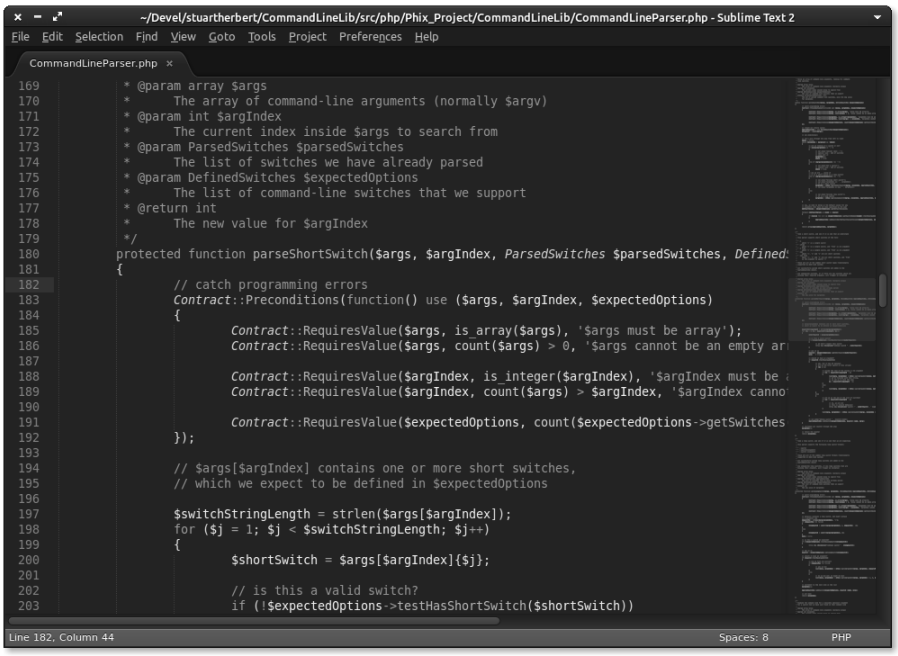


Imagen 22 Captura de Sublime Text ejecutándose en una distribución de GNU / Linux

De entre los entornos de desarrollo vistos, elegimos **Netbeans** por su parecido con Eclipse (el utilizado como herramienta de aprendizaje en el grado en ingeniería informática), su uso más eficiente de los recursos, su soporte a extensiones que enriquecen la funcionalidad del entorno y sus paquetes de instalación específicos con soporte para las tecnologías que usaremos en nuestro proyecto. De esta forma garantizamos una baja curva de aprendizaje a la vez que disponemos de un entorno potente y fiable para desarrollar.

Una vez instalado el *I.D.E.* en nuestro computador, deberemos preparar el despliegue de la aplicación. Recordemos que estamos desarrollando un *software* cuya arquitectura se basa en el modelo cliente-servidor, por lo que es necesario instalar un servidor web en el que alojar la aplicación y un agente de usuario (navegador web) que haga el papel de cliente al realizar una solicitud al servidor. Al recibir la petición, el servidor pondrá en marcha el intérprete PHP (necesario para nuestro caso) que ejecutará la lógica de negocio, construirá la interfaz de la aplicación adecuada para el cliente y le enviará todos los archivos necesarios para su correcta visualización.

Para conseguirlo, usaremos un paquete de *software* web libre y multiplataforma que incluye las tecnologías más comunes en el despliegue de una aplicación web: servidor web Apache [25], base de datos MySQL y un intérprete PHP. Hay diferentes soluciones que cubren nuestra necesidad, como **AppServ**, **WAMP** o **XAMPP** [27] (**Imagen 23**).

En nuestro caso, seleccionamos XAMPP (acrónimo de: X, por ser multiplataforma; Apache; MySQL; PHP y Python) por su sencillez y rapidez en la instalación y configuración y por poder discriminar qué paquetes estarán disponibles tras la instalación. Por ejemplo, podríamos elegir no instalar el lenguaje Python en nuestro sistema si no vamos a usarlo, como es nuestro caso.

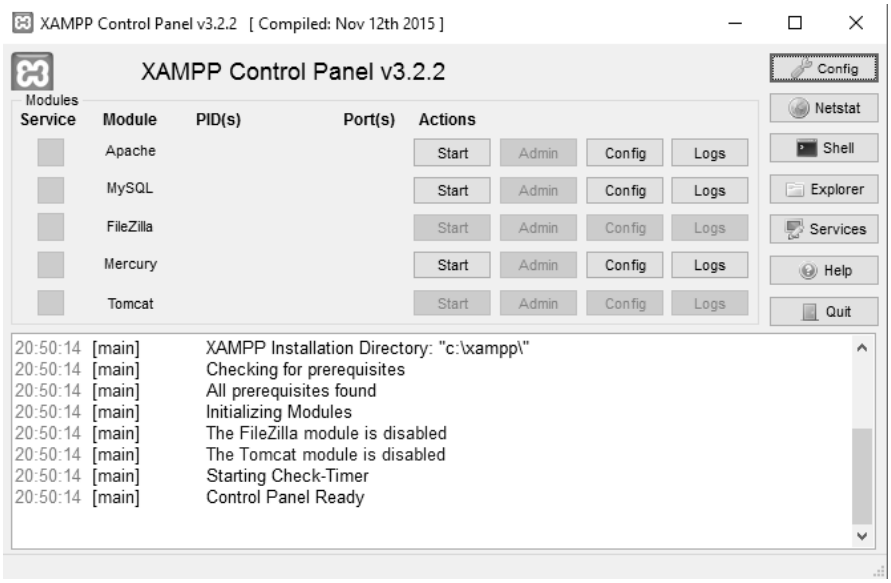


Imagen 23 Panel de control de XAMPP

La instalación incluye el servidor web Apache con un intérprete PHP y un servidor de base de datos MySQL, ambas herramientas imprescindibles para desplegar nuestra aplicación web.

Es posible configurar Netbeans para que, cada vez que se abra el proyecto en el entorno de desarrollo, se copien los archivos fuente en la carpeta de publicación del servidor Apache (**Imagen 24**).

También deberemos limitar el acceso a la web a máquinas en nuestra red local mediante un cortafuego para evitar accesos no deseados, aunque en el caso de estar conectados a la red tras un dispositivo *N.A.T.* (*Network Address Translator*), no habría posibilidad de acceder a nuestra web sin aplicar un reenvío de puertos a la máquina local en la que está funcionando el servidor web.

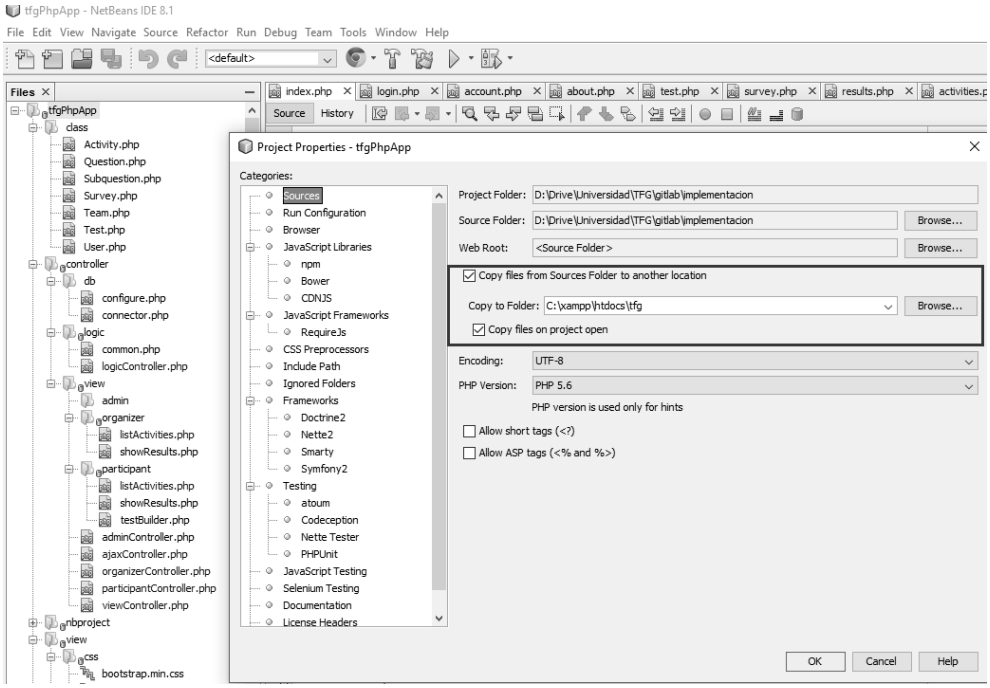


Imagen 24 Ajuste en Netbeans para copiar el código fuente a la carpeta de publicación del servidor web

Tras tener configurado el entorno en el que desarrollaremos y testearemos la aplicación, veremos las librerías que necesitaremos para completar las funcionalidades que especificamos en el análisis inicial.

Como vimos en el apartado perteneciente al modelado de la solución planteada, crearemos una página especial en la que se mostrarán diversos datos relativos a estadísticas sobre la distribución de roles y personalidades dentro de un grupo o actividad y, además, cada usuario podrá ver en una página personal la valoración que le han otorgado sus compañeros de equipo.

Para mostrar esta información, recurrimos a una librería que permite generar gráficas (objetos), pudiendo personalizarlas con nuestros propios datos e incluso modificar su forma y color, adaptándola a nuestras necesidades. Esta librería, de nombre ***FusionCharts***, está escrita en PHP y preparada para procesar datos obtenidos de una base de datos, por lo que el coste de la integración en nuestro sistema es muy bajo [28].

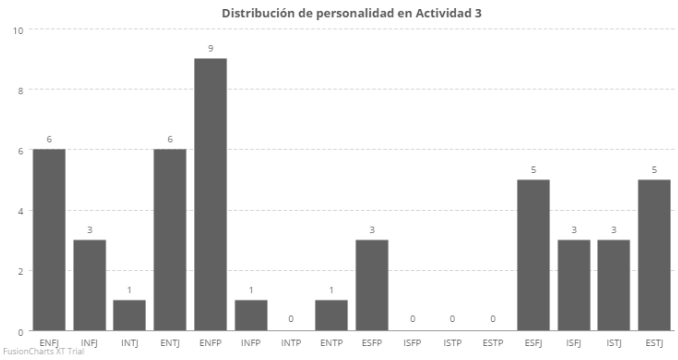


Imagen 25

Ejemplo de gráfica generada mediante *FusionCharts* y visualizada en un navegador web

Por otro lado, aunque en el diseño de la interfaz no es conveniente colocar demasiados elementos gráficos en la pantalla, sí que es recomendable usar adecuadamente una iconografía que permita al usuario reconocer rápidamente acciones y lugares.

Por ello, y para prescindir de imágenes que ocupan demasiada información redundante para nuestro fin, aprovecharemos librerías basadas en **fuentes tipográficas** que asocian a cada caracter, no solo letras o símbolos, sino también iconos.

Su forma de integración en nuestra página se basa en el enlazado de archivos tipográficos *.otf*, *.eot*, *.svg*, *.ttf* o *.woff* desde una hoja de estilos *.css*. Después, basta con aplicar unas clases concretas a etiquetas *html* determinadas, y obtendremos de resultado visible una amplia variedad de iconografía, incluso con personalización de tamaño, que hará más usable y visualmente agradable a nuestra aplicación.

Hay varias librerías que ofrecen soluciones similares, como *FontAwesome* [29] o *Glyphicons*, diferenciándose principalmente en el número de iconos disponibles y sus aspectos. Éstas nos permitirán extender la experiencia visual de la página.

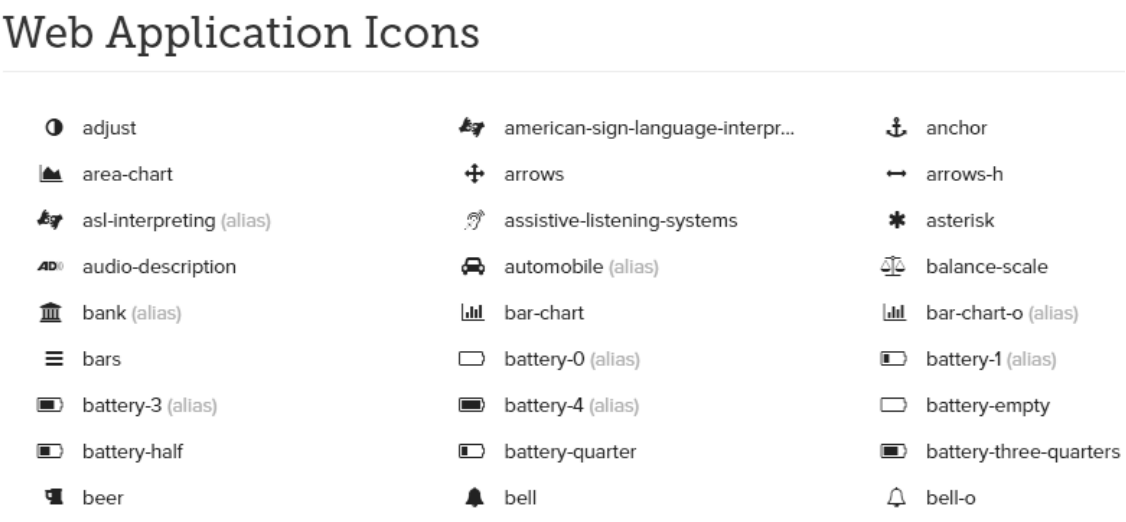


Imagen 26 Ejemplo de iconos de *FontAwesome* preparados específicamente para uso en aplicaciones *web*

También podemos hacer uso de plantillas o temas para el *framework Bootstrap* que utilizaremos para construir la interfaz de la aplicación. En concreto, existen plantillas gratuitas que permiten adaptar la interfaz a un modelo de sitio *web* determinado, como por ejemplo un panel de administración o una aplicación *web*.

Podemos encontrar muchos ejemplos en la página oficial de *Bootstrap* https://startbootstrap.com

La plantilla que vamos a escoger nosotros está pensada principalmente para aplicaciones *web* y paneles de administración: ***SB Admin 2*** [30] (**Imagen 27**). Dispone de una interfaz similar a la que nosotros buscamos, con un menú de acciones principal en la parte superior, otro menú de navegación en el lateral izquierdo y una zona grande y destacada en las partes central y derecha de la página, ocupando la mayor superficie y dando el papel principal al contenido.

Imagen 27 Tema *SB Admin 2* basado en *Bootstrap* funcionando en un navegador, con datos de muestra

Esta plantilla es fácilmente integrable en nuestra página realizando las siguientes acciones:

- Enlazar las librerías *.css* y *.js* del tema a la cabecera *HTML*.
- Utilizar la estructura y clases definidas en el tema en nuestra página para conseguir la apariencia deseada. Toda la información está disponible en la documentación aportada tanto en su página como en los archivos fuente.

Al estar basada en *Bootstrap*, la sintaxis es la misma, incluyendo nuevas clases predefinidas que permiten mostrar fácilmente distintos elementos ya diseñados.

Por último, vamos a ver una extensión de *jQuery* que permite validar formularios de forma sencilla y visualmente más agradable desde el lado del cliente, sin

requerir a complejos anidamientos de condiciones en el código fuente o la recarga o redireccionamiento de páginas en la aplicación, lo que provocaría frustración en el usuario y sería un síntoma de escasa usabilidad.

Además, incluye algoritmos de validación concretos para comprobar campos pertenecientes a valores de correo electrónico, contraseñas, enlaces e incluso tarjetas bancarias.

Este *plugin* también incluye un sistema para mostrar mensajes de error en cada campo no validado de un formulario, incluso en diferentes idiomas, a elegir por el desarrollador. Esto permite una integración rápida y poco costosa de un sistema de validación de formularios.

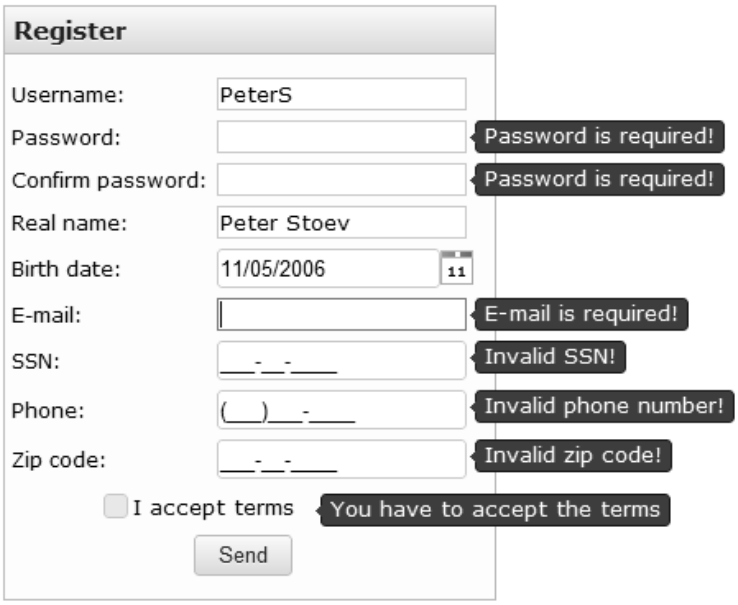


Imagen 28 Ejemplo de validación de un formulario con *jQuery Validation Plugin*

Tras ver las librerías que usaremos en nuestra aplicación, ha llegado el momento de ponerse manos a la obra y realizar la implementación de nuestra solución. Dada la naturaleza del proyecto y la variedad de funcionalidades, desarrollaremos las principales características de éste y obtendremos una versión limitada, pero con el potencial y la estructura necesaria para continuar su desarrollo e incluso ampliarlo posteriormente.

2.3 Implementación

Como vimos en el apartado de arquitectura de *software*, nuestra aplicación estará basada en el patrón **Modelo-Vista-Controlador**. Esto requerirá estructura el programa en tres secciones principales interconectadas.

En primer lugar crearemos el **Modelo**: ficheros *.php* con las clases, atributos y métodos que definimos anteriormente. Estos se encuentran dentro de la carpeta *class* del proyecto.

Seguidamente, implementaremos la interfaz que concretamos anteriormente: la **Vista**. Los ficheros relacionados (de los tipos *.php*, *.html*, *.js*, *.css*, *.jpg*, *.png*, entre otros), estarán bajo el directorio *view*. Estos ficheros definirán la estructura y estilo de las páginas diseñadas para nuestra aplicación.

Por último, crearemos los **Controladores** necesarios que, con ayuda de las clases del Modelo, construirán la Vista, controlarán el flujo del programa y servirán de puente para guardar y obtener información de la base de datos. Por ello, los controladores estarán agrupados por su función: controlador de base de datos, de lógica o de vista.

Dada la naturaleza del proyecto (aplicación web), necesita un fichero principal que sirva de entrada a cualquier usuario. Éste fichero, *index.php*, ubicado en el directorio raíz, controlará el acceso a la página y lanzará el controlador adecuado según el tipo de usuario que haya accedido. A éste fichero lo denominaremos *dispatcher*, por su función de encaminamiento.

También incluiremos en el directorio raíz un fichero, llamado *constants.php*, que sirva para definir constantes con las rutas absolutas a los distintos directorios de nuestra aplicación. Esto nos servirá para evitar problemas con enlaces relativos desde ficheros en diferentes directorios, así como prevenir un mayor mantenimiento del código en caso de que el proyecto se mueva a otros servidores con otros entornos.

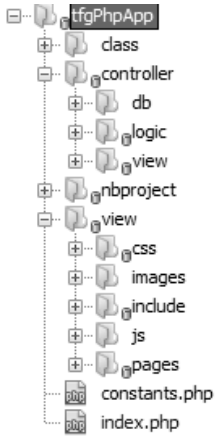


Imagen 29 Árbol de directorios de la aplicación

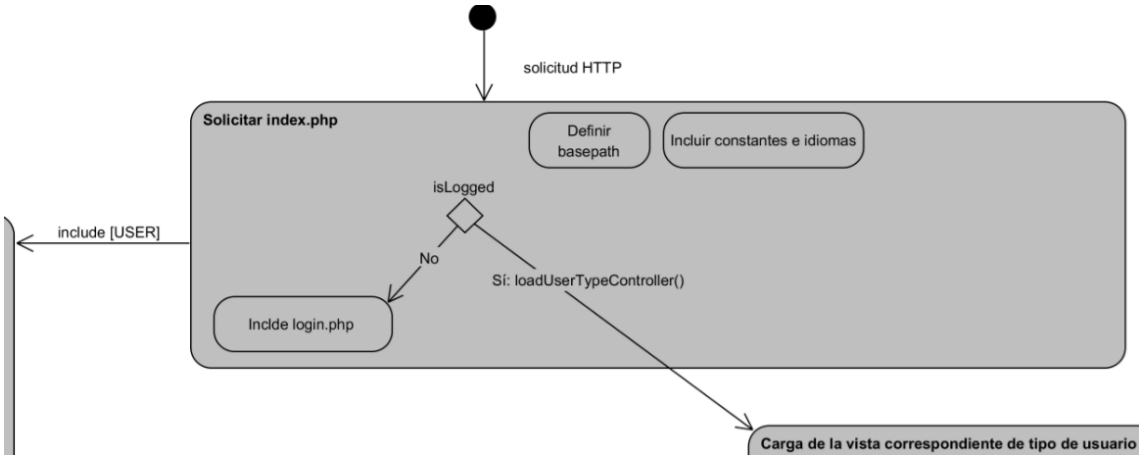


Imagen 30 Flujo de la aplicación al solicitar el fichero principal *index.php*

Cuando el usuario solicita acceder a la *web*, se comprueba que entra con unos datos correctos o se le mostrará un formulario de registro. Si todo va bien, se le redirigirá al controlador adecuado, el cual cargará, si se ha especificado, la página concreta solicitada (**Imagen 30**). La página, localizada en un fichero *.php* dentro del directorio *view/pages*, contiene la estructura básica de la vista, desarrollada con *HTML*, *CSS*, *Javascript* y *Bootstrap*. Además, hace las llamadas correspondientes al controlador de vista (que a su vez se comunica con el controlador de la lógica), que recuperará los datos necesarios y montará dinámicamente en *HTML* la información obtenida (**Imagen 31**).

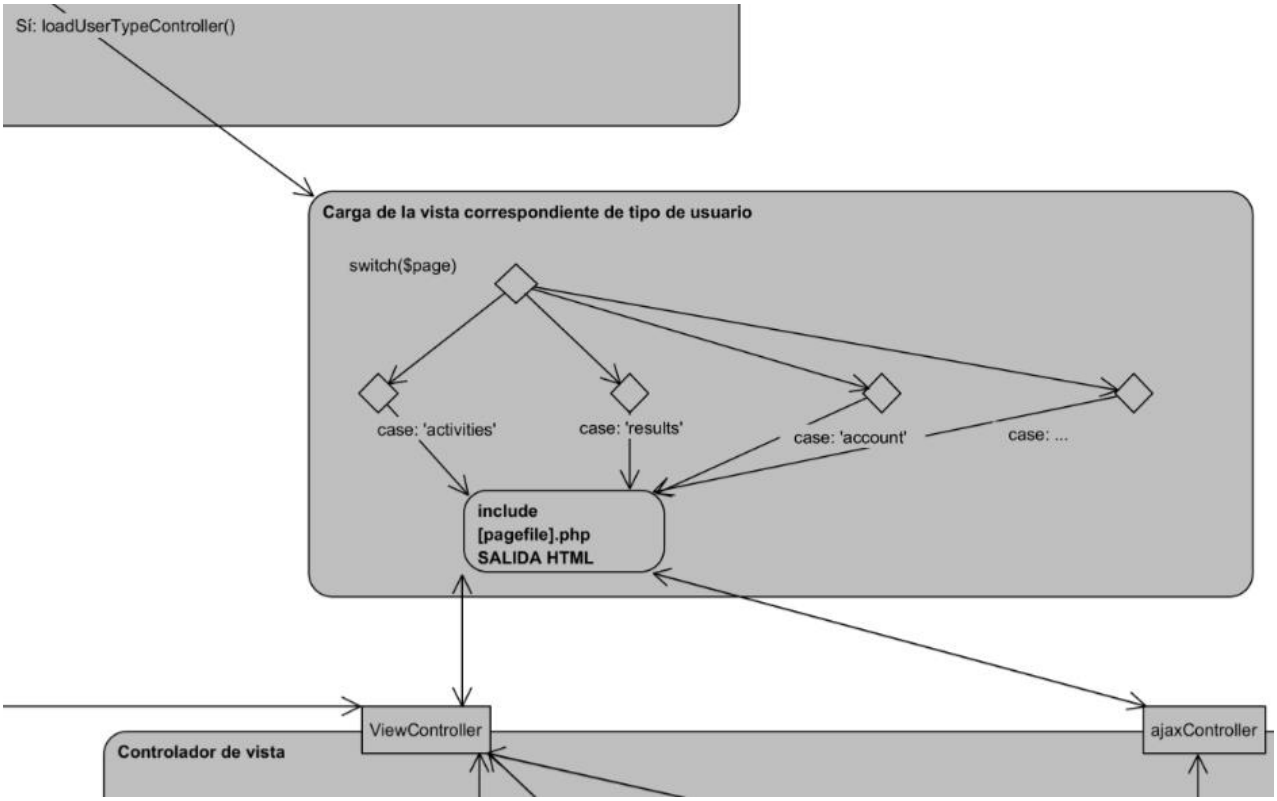


Imagen 31 Carga de la página correspondiente y sus llamadas al controlador de Vista para generarse dinámicamente

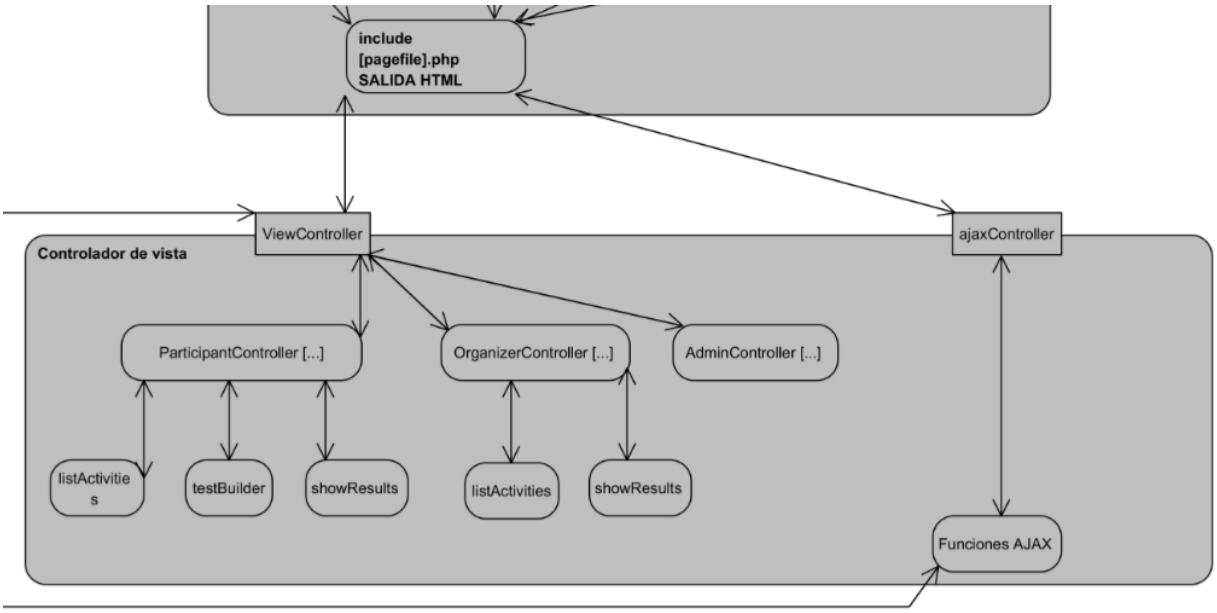


Imagen 32 Flujo de información en el controlador de vista

Del controlador de vista (**Imagen 32**), *ViewController*, heredan otros tres controladores específicos para cada tipo de usuario: *ParticipantController*, *OrganizerController* y *AdminController*. *ViewController* incluye funciones comunes, como mostrar información de la cuenta de un usuario.

Los controladores específicos contienen métodos usados únicamente por cada tipo de usuario, ya tengan objetivos totalmente diferentes o similares, pero con datos obtenidos de forma distinta. Estos controladores construirán parte de la interfaz con datos obtenidos dinámicamente de base de datos.

De forma parecida, creamos un nuevo controlador de vista, llamado *ajaxController*, que es el encargado de reunir todos los métodos que serán llamados, desde la interfaz ya construida, por un usuario al interactuar con la página. Además, este controlador proporcionará los datos obtenidos dinámicamente o actualizará el sistema con los nuevos datos introducidos, mediante la tecnología *AJAX*.

Por otra parte, el controlador de lógica, *logicController*, recoge las funciones y algoritmos necesarios para administrar, validar, guardar y obtener toda la información de nuestra sistema teniendo en cuenta los requisitos de la aplicación. El controlador usa las clases modeladas para procesar los datos de forma lógica, lo cual permite mayor legibilidad y mejor mantenimiento del programa (paradigma de la programación orientada a objetos) (**Imagen 33**).

También se encargará de realizar las llamadas oportunas a la base de datos, ya sea para almacenar o para obtener información de ella, interactuando con el controlador de base de datos *connector.php* (**Imagen 34**).

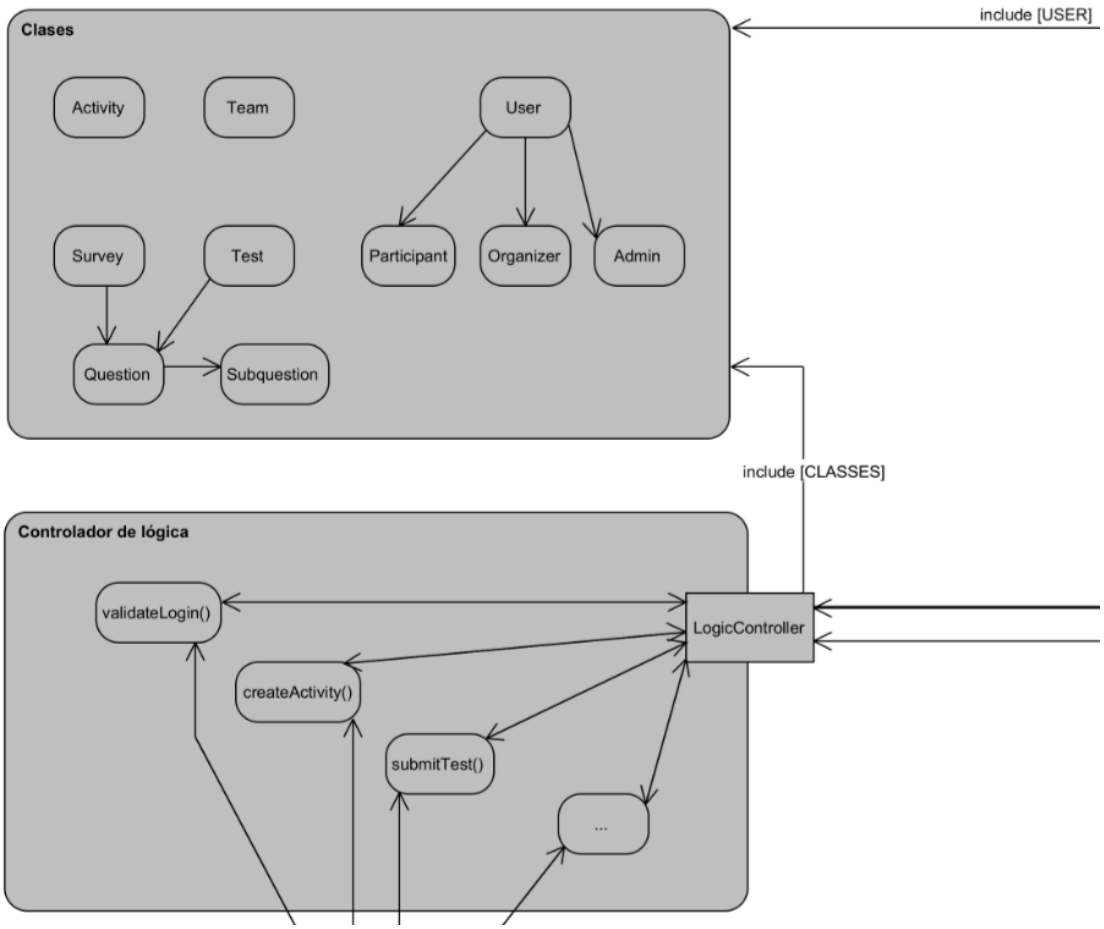


Imagen 33 Flujo de información en el controlador de la lógica

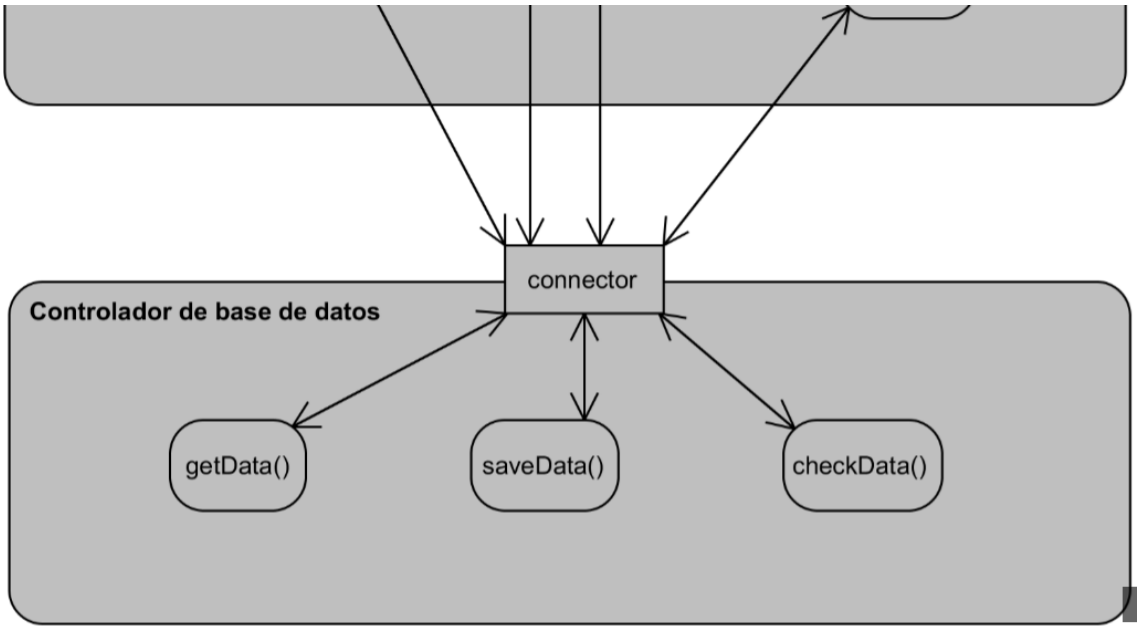


Imagen 34 Funcionamiento lógico del controlador de la base de datos

El índice de métodos del **controlador de lógica** se muestra a continuación:

Métodos relativos al acceso y registro en la aplicación

- **validateLogin**($email, $password)
  Valida el acceso al sistema.
- **registerInvitated**($token, $name, $surnames, $dni, $registerPassword, $city, $country)
  Registra en el sistema un usuario invitado previamente.
- **inviteParticipant**($activityId, $email)
  Envía una invitación de registro a una dirección de correo electrónico suministrada. Esta invitación está asociada a una actividad concreta.

Métodos relativos a la gestión de usuarios

- **getUserInfo**()
  Obtiene la información del usuario actualmente conectado al sistema.
- **getUser**($searchTerm)
  Obtiene un objeto Usuario según un término de búsqueda.
- **editUser**($user)
  Modifica un usuario, pasado como parámetro, en el sistema.
- **editUserPass**($userid, $pass)
  Actualiza la contraseña de un usuario.
- **getParticipant**($searchTerm)
  Obtiene un objeto Participante, según un término de búsqueda.
- **getParticipants**($activityId)
  Obtiene todos los participantes de una actividad.
- **getNewParticipants**($activityId)
  Obtiene todos los usuarios actualmente invitados a una actividad.
- **hasAcceptedActivity**($userId, $activityId)
  Comprueba si un usuario ha aceptado la invitación a una actividad.
- **hasDoneTest**($testId, $userId)
  Comprueba si un usuario ha realizado un test concreto.
- **hasDoneAllRequiredTests**($userId, $activityId)
  Comprueba si un usuario ha realizado todos los tests requeridos en una actividad.

Métodos relativos a la gestión de actividades

- **createActivity**($activityName, $shortName, $category, $visible, $startDate, $finishDate, $description, $minTeamSize, $maxTeamSize, $showEmails, $requiredTestIds, $organizerId)
  Crea en el sistema una nueva actividad.
- **updateActivity**($activityId, $organizerId, $activityName, $shortName, $category, $visible, $startDate, $finishDate, $description, $minTeamSize, $maxTeamSize, $showEmails, $requiredTests)
  Actualiza una actividad existente.
- **getActivity**($activityId)
  Obtiene un objeto Actividad.
- **getActivities**($organizerId)
  Obtiene un *array* de Actividades administradas por un organizador determinado.
- **startActivity**($activityId)
  Marca una actividad como iniciada.
- **finishActivity**($activityId)
  Marca una actividad como finalizada.

- **getNewActivitiesByOrganizer**($organizerId)
  Obtiene actividades aún no iniciadas y administradas por un organizador.
- **getOnGoingActivitiesByOrganizer**($organizerId)
  Obtiene actividades iniciadas pero no finalizadas, administradas por un organizador.
- **getEndedActivitiesByOrganizer**($organizerId)
  Obtiene actividades finalizadas, administradas por un organizador.
- **getNewActivitiesByParticipant**($participantId)
  Obtiene las actividades aún no iniciadas a las que ha sido invitado.
- **getOnGoingActivitiesByParticipant**($participantId)
  Obtiene las actividades en curso a las que ha sido invitado.
- **getEndedActivitiesByParticipant**($participantId)
  Obtiene actividades finalizadas a las que fue invitado.

Métodos relativos a la gestión de datos de actividades

- **joinActivity**($activityId, $userId)
  Una vez aceptada una invitación, incluye al usuario como participante.
- **kickParticipant**($activityId, $userId)
  Expulsa un participante de una actividad concreta.

- **getTeams**($activityId)
  Obtiene los equipos de una actividad.
- **getTeamId**($activityId, $userId)
  Obtiene el identificador de un equipo al que pertenece un usuario, según la actividad.
- **sendSurveys**($activityId)
  Marca una actividad como permitida para rellenar su encuesta de satisfacción.
- **getRequiredTests**($activityId)
  Obtiene los objetos Test requeridos en una actividad concreta.

Métodos relativos a la formación de equipos

- **doTeams**($activityId)
  Realiza la llamada al algoritmo generador de equipos y crea los equipos asociados a una actividad.
- **undoTeams**($activityId)
  Deshace los equipos y vuelve a su estado anterior.

Métodos relativos a la gestión de tests y encuestas

- **getTest**($testid)
  Obtiene un objeto Test.
- **submitTest**($test, $userId)
  Guarda los resultados de un objeto Test, asociados a un usuario.
- **getSurvey**($surveyid)
  Obtiene un objeto Encuesta, según su identificador.
- **submitSurvey**($survey, $activityId, $userId)
  Almacena los datos de una encuesta de satisfacción, según la actividad y el usuario que la rellenó.
- **getBELBINdata**()
  Obtiene los datos específicos del test *Belbin*.
- **getMBTIdata**()
  Obtiene los datos específicos del test *MBTI*.
- **getAvailableTests**()
  Obtiene los tests existentes y disponibles en la aplicación.

Métodos generales

- **checkCoincidentEmails**($search)
  Devuelve los correos electrónicos que coinciden con el término buscado.

Respecto al controlador de la base de datos, destacamos los tres métodos principales:

- **query**($sql)
  Realiza la consulta, pasada como parámetro, en la base de datos.
- **insertAndGetId**($sql)
  Realiza la consulta de inserción, parametrizada, y devuelve el identificador de la última fila insertada.
- **updateAndGetUpdatedRows**($sql)
  Devuelve el número de filas afectadas por la consulta de actualización realizada.

En cuanto a los controladores de vista, tenemos:

- *ViewController*
  - **showNoNotification**()
    Muestra un bloque *HTML* con un mensaje que indica que no hay notificaciones nueva.
  - **showAccountData**($userid)
    Muestra el formulario con la información de la cuenta personal.

- *ParticipantController*
  - **listActivitiesNew**()
    Muestra las actividades no iniciadas.
  - **listActivitiesOngoing**()
    Muestra las actividades en curso.
  - **listActivitiesEnded**()
    Muestra las actividades finalizadas.

  - **listRequiredTests**($activityId)
    Muestra los tests requeridos.
  - **showPersonalResults**($userId)
    Muestra en gráficas las estadísticas del usuario.

  - **loadTest**($testId, $userId)
    Carga el test correspondiente, comprobando que no ha sido ya realizado.
  - **loadSurvey**($activityId, $userId)
    Carga la encuesta correspondiente.

- *OrganizerController*
  - **listRequiredTests**($activityId)
  - **listActivitiesNew**()
  - **listActivitiesOngoing**()
  - **listActivitiesEnded**()
  - **createActivity**()
    Muestra el formulario para crear una nueva actividad.
  - **editActivity**($activityId)
    Muestra el formulario para modificar una actividad.
  - **showActivityResults**($organizerId)
    Muestra en gráficas los resultados estadísticos de una actividad.

- *AjaxController* (envía los datos al controlador de lógica y espera una respuesta, satisfactoria o no)
  - **validateLogin**($email, $password)
  - **registerInvitated**($token, $name, $surnames, $dni, $registerPassword, $city, $country)
  - **modifyUserPass**($userId, $modifyPass)
  - **modifyUser**($userId, $modifyEmail, $modifyName, $modifySurnames, $modifyDni, $modifyCity, $modifyCountry)
  - **startActivity**($activityId)
  - **finishActivity**($activityId)
  - **createActivity**($activityName, $shortName, $category, $visible, $startDate, $finishDate, $description, $minTeamSize, $maxTeamSize, $showEmails, $requiredTests, $organizerId)
  - **updateActivity**($activityId, $organizerId, $activityName, $shortName, $category, $visible, $startDate, $finishDate, $description, $minTeamSize, $maxTeamSize, $showEmails, $requiredTests)
  - **joinActivity**($activityId, $userId)
  - **inviteParticipant**($activityId, $modifyEmail)
  - **checkCoincidentEmails**($search)
  - **kickParticipant**($activityId, $userId)
  - **doTeams**($activityId)
  - **undoTeams**($activityId)

    - **submitTest**($test, $userId)
    - **sendSurveys**($activityId)

Para generar las vistas, tenemos enlazadas a cada página *HTML* ficheros externos con los estilos y funciones requeridos para conseguir la interfaz, animaciones y funcionalidades deseadas (los ficheros en **negrita** son de creación propia):

- *view*
    - *css*
        - bootstrap.min.css
        - font-awesome.min.css
        - sb-admin-2.css
        - **theme-responsive.css** (adaptado al tamaño de pantalla)
        - **theme.css**
    - *include*
        - *fonts*
        - *languages*
        - **footer.php**
        - fusioncharts.php
        - **header.php**
        - **nav.php**
        - **search.php**
        - **sideMenu.php**
    - *js*
        - *fusioncharts*
        - *localization*
        - additional-methods.min.js
        - bootstrap.min.js
        - **functions.php** (llamadas AJAX)
        - jquery-2.2.0.min.js
        - jquery.validate.min.js
        - metisMenu.min.js
        - sb-admin-2.js
        - **theme.php** (animaciones)
    - *pages*
        - *organizer*
            - **activities.php**
            - **results.php**
        - *participant*
            - **activities.php**

- **results.php**
- **survey.php**
- **test.php**

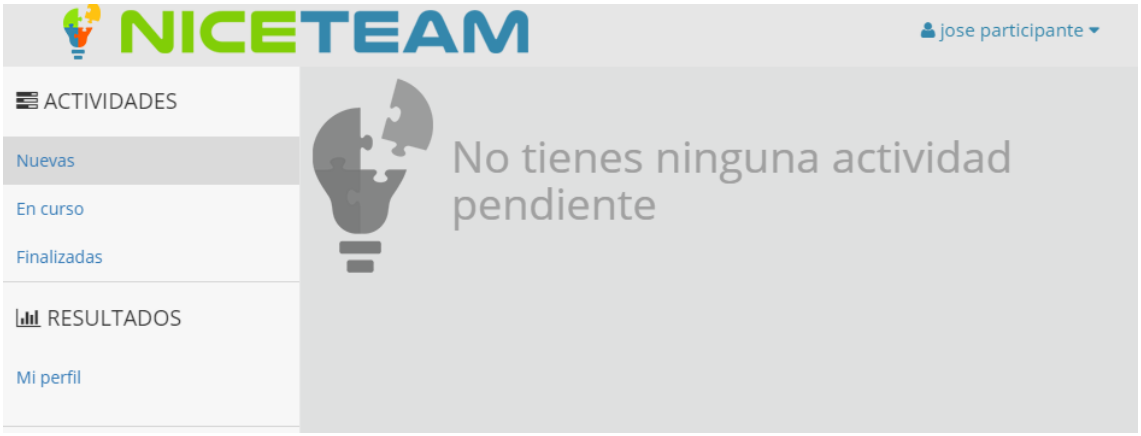
- **about.php**
- **account.php**
- **login.php**
- **logout.php**

Tras integrar adecudamente la interfaz diseñada con las tecnologías y librerías utilizadas, así como implementar correctamente los métodos que permitirán el funcionamiento deseado de la lógica (todo ello enlazado mediante los controladores correspondientes), obtenemos el siguiente resultado:

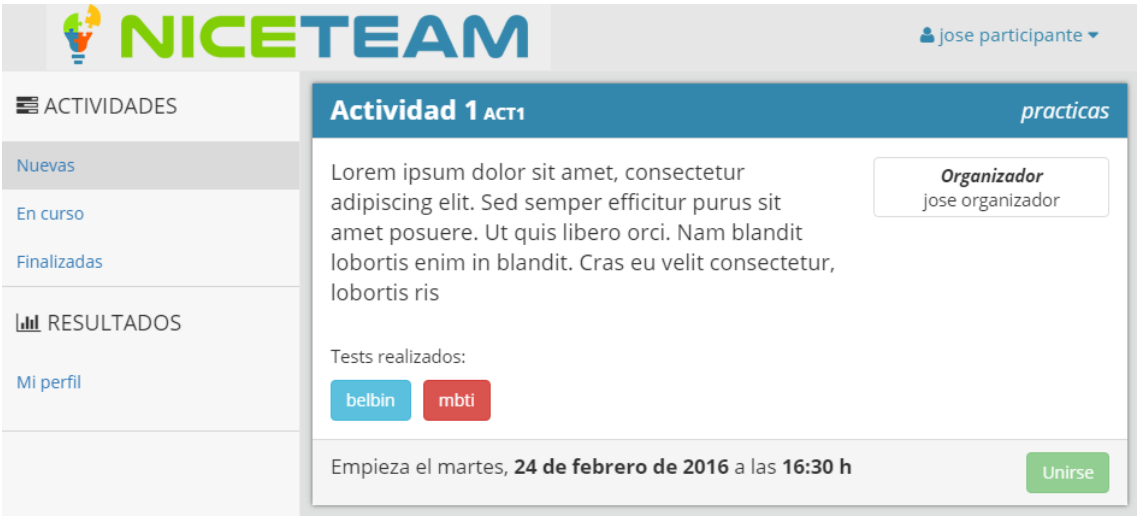
Página de inicio de sesión o registro con invitación

Página de inicio común

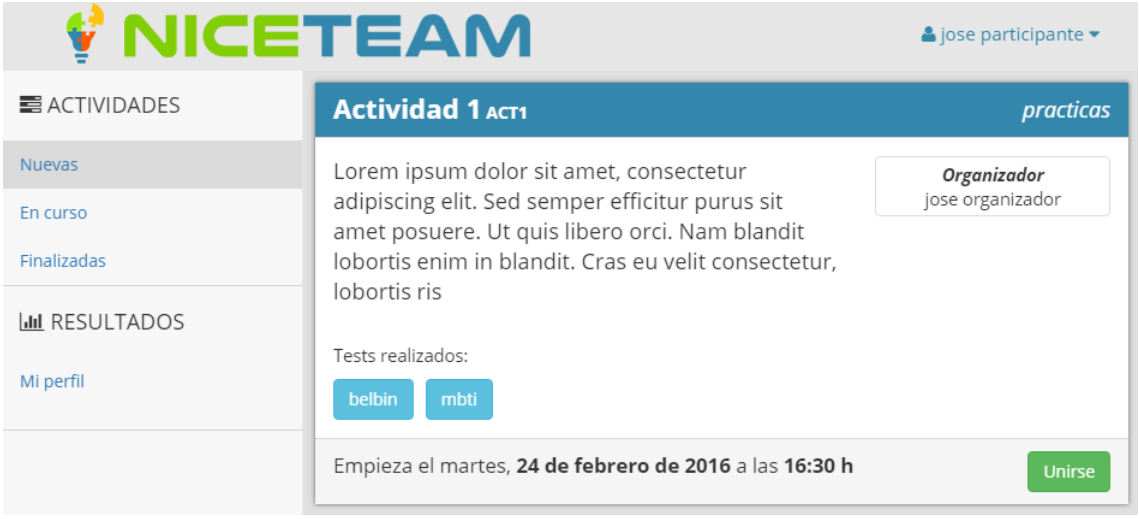
Páginas de nuevas actividades de un participante



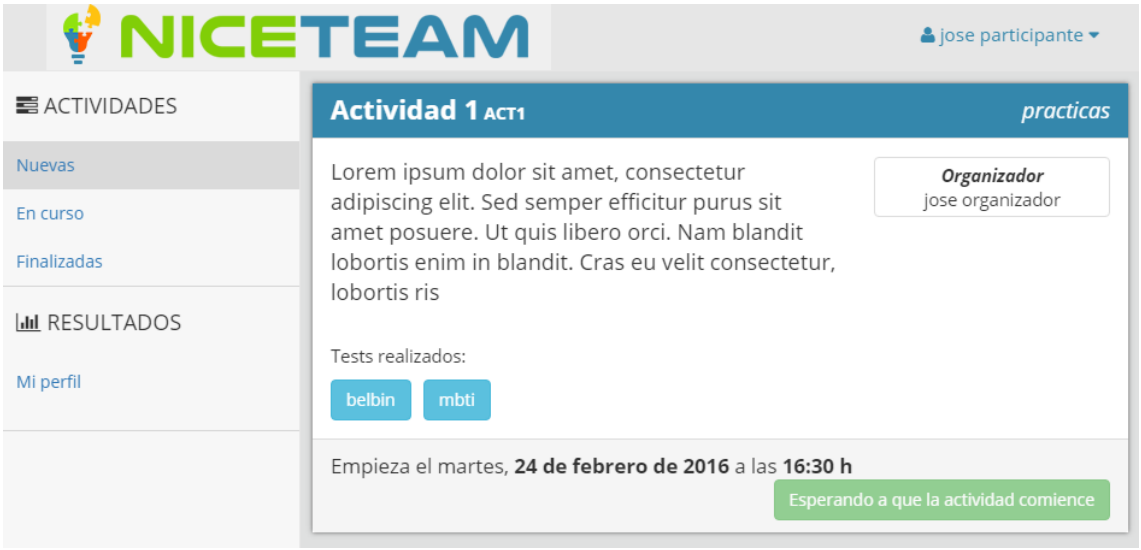
Participante sin actividades nuevas disponibles



Actividad nueva disponible con uno de los tests rellenado

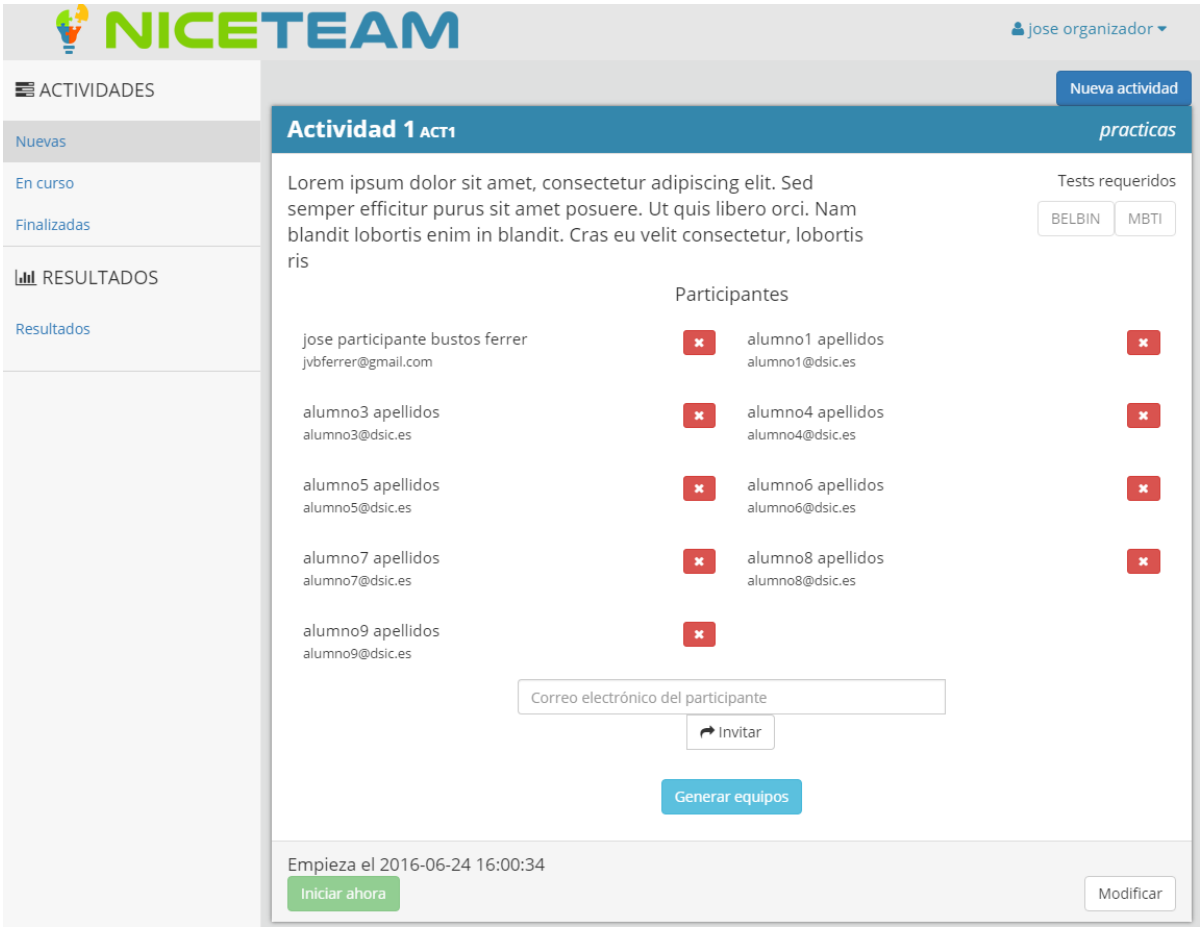


Actividad nueva disponible con todos los tests requeridos completados

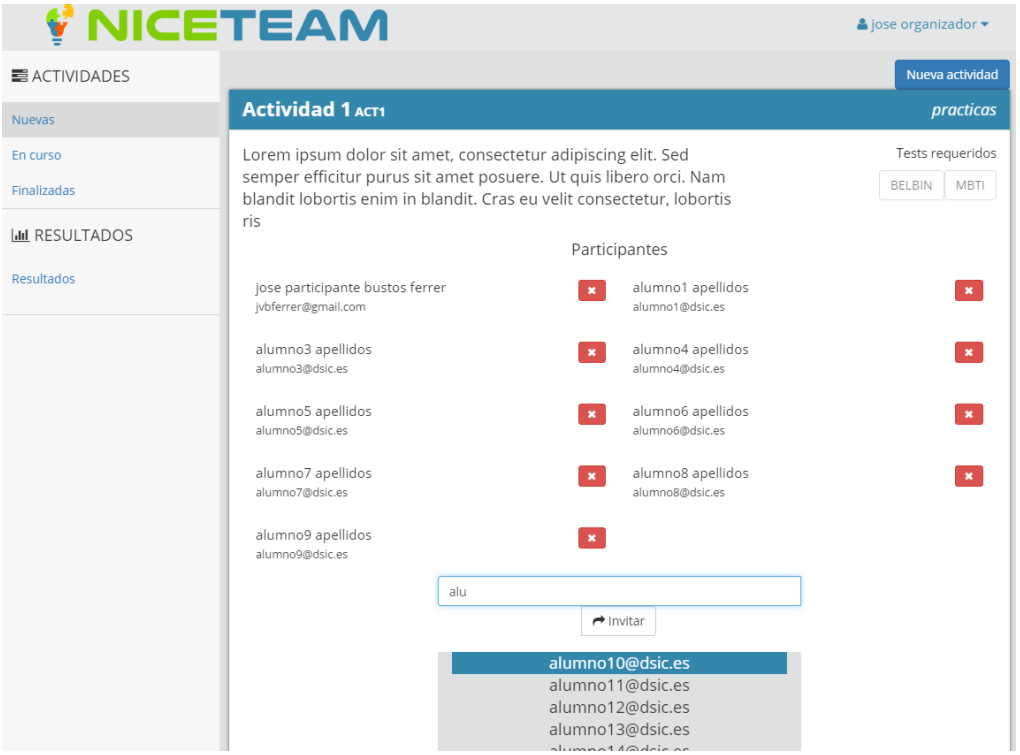


Actividad nueva a la que ya se ha unido el Participante

Páginas de nuevas actividades de un organizador

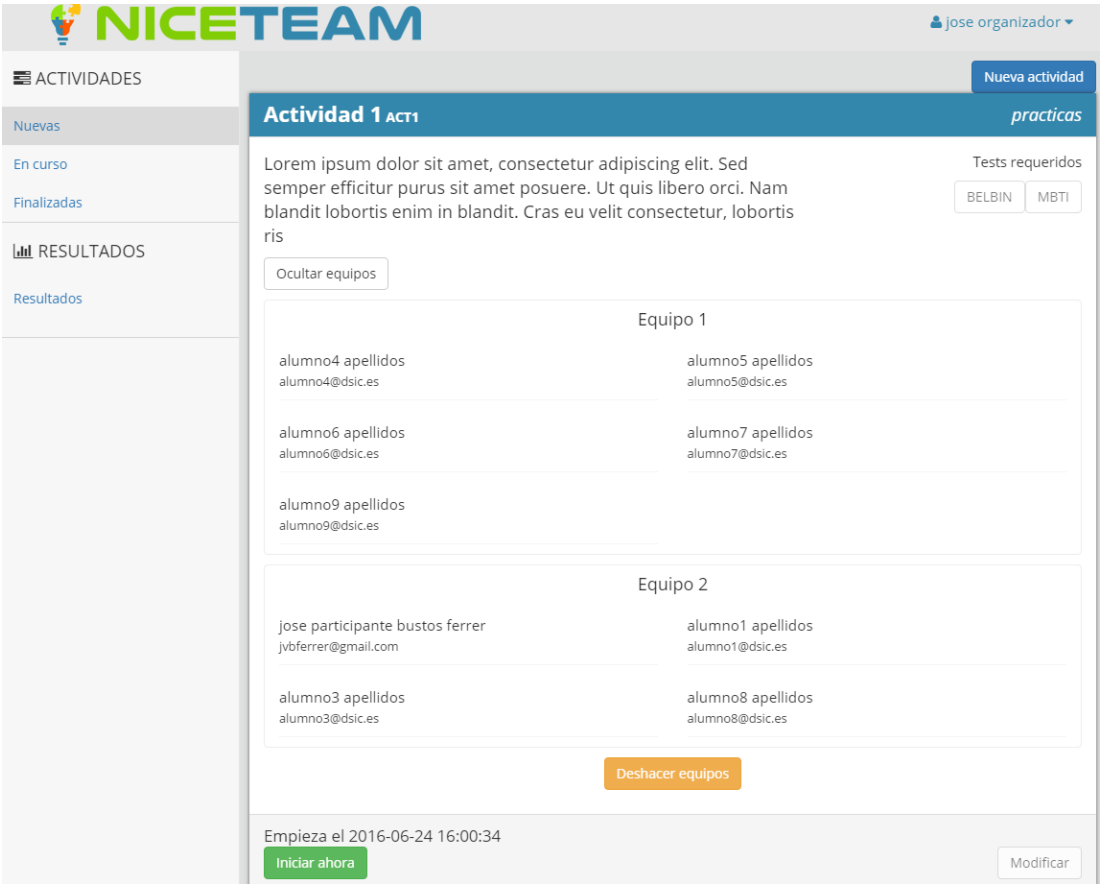


Vista de los participantes unidos a la Actividad. Desde aquí también se crean nuevas Actividades



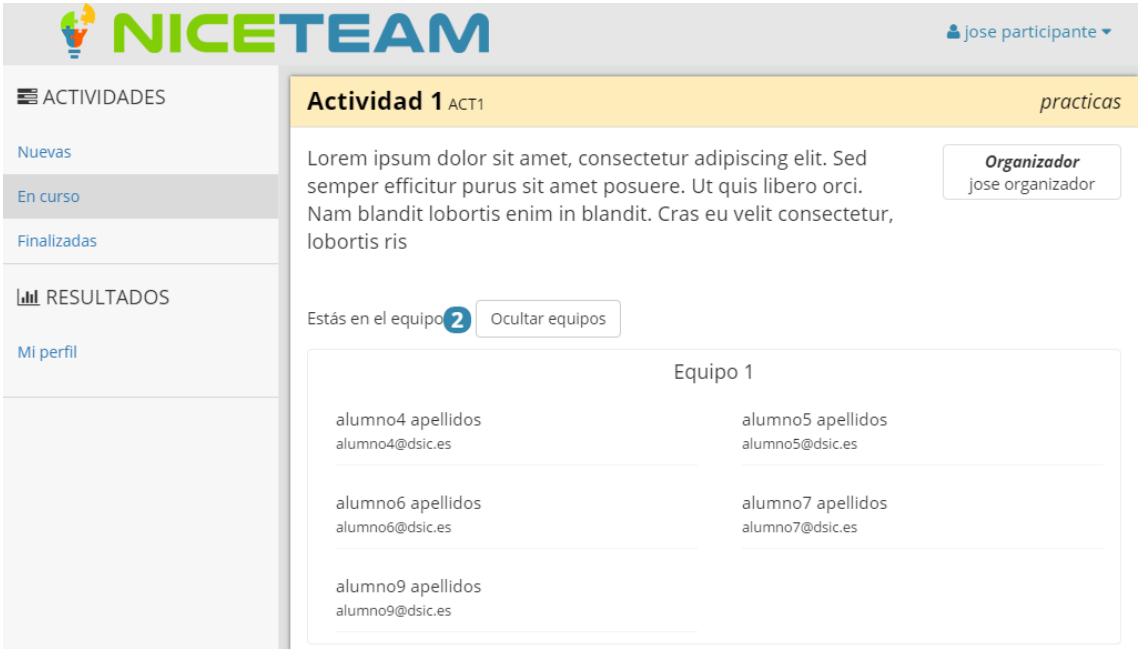
Formulario *AJAX* para invitar nuevos participantes

Páginas de nuevas actividades de un organizador con equipos generados



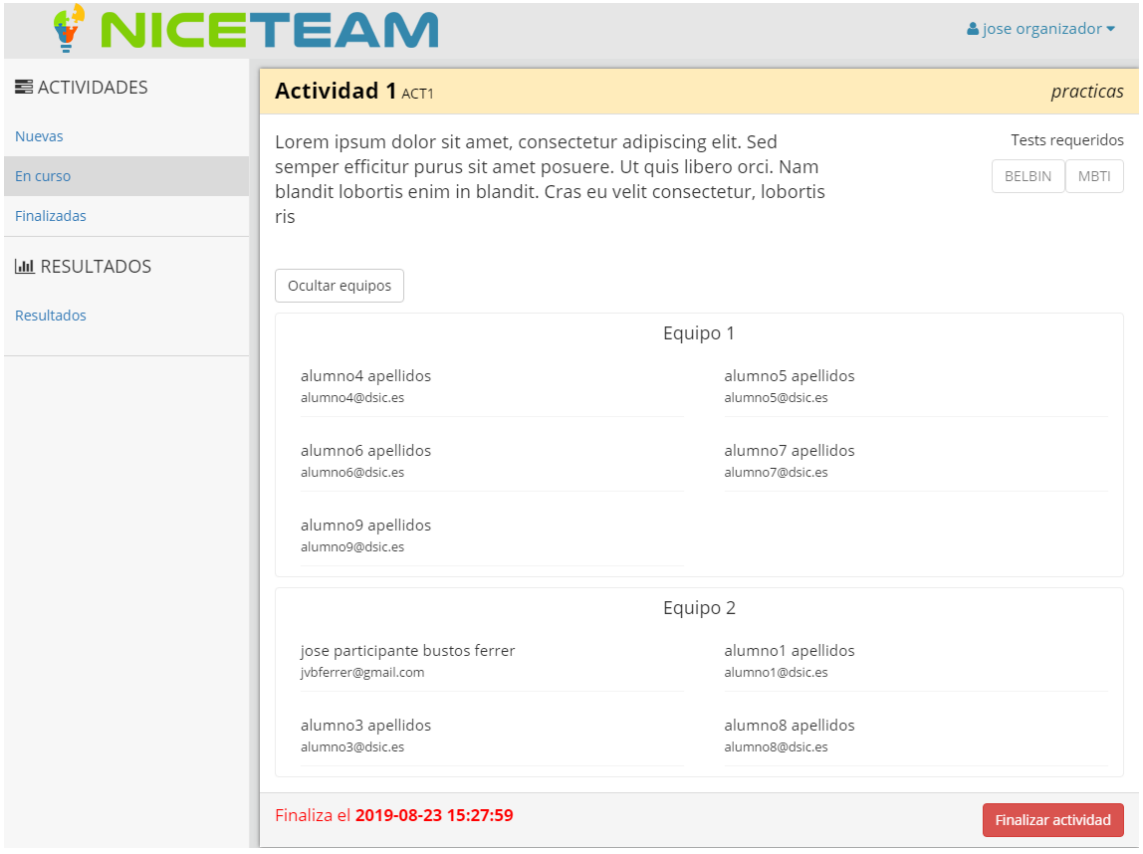
Actividad no iniciada con los equipos generados

Páginas de actividades en curso de un participante



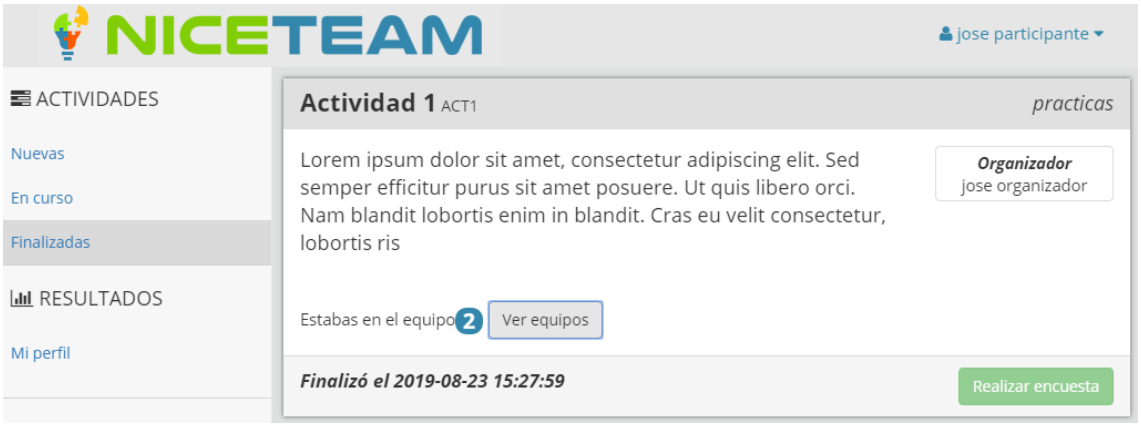
Actividad en curso, con posibilidad de ver a los compañeros

Páginas de actividades en curso de un organizador



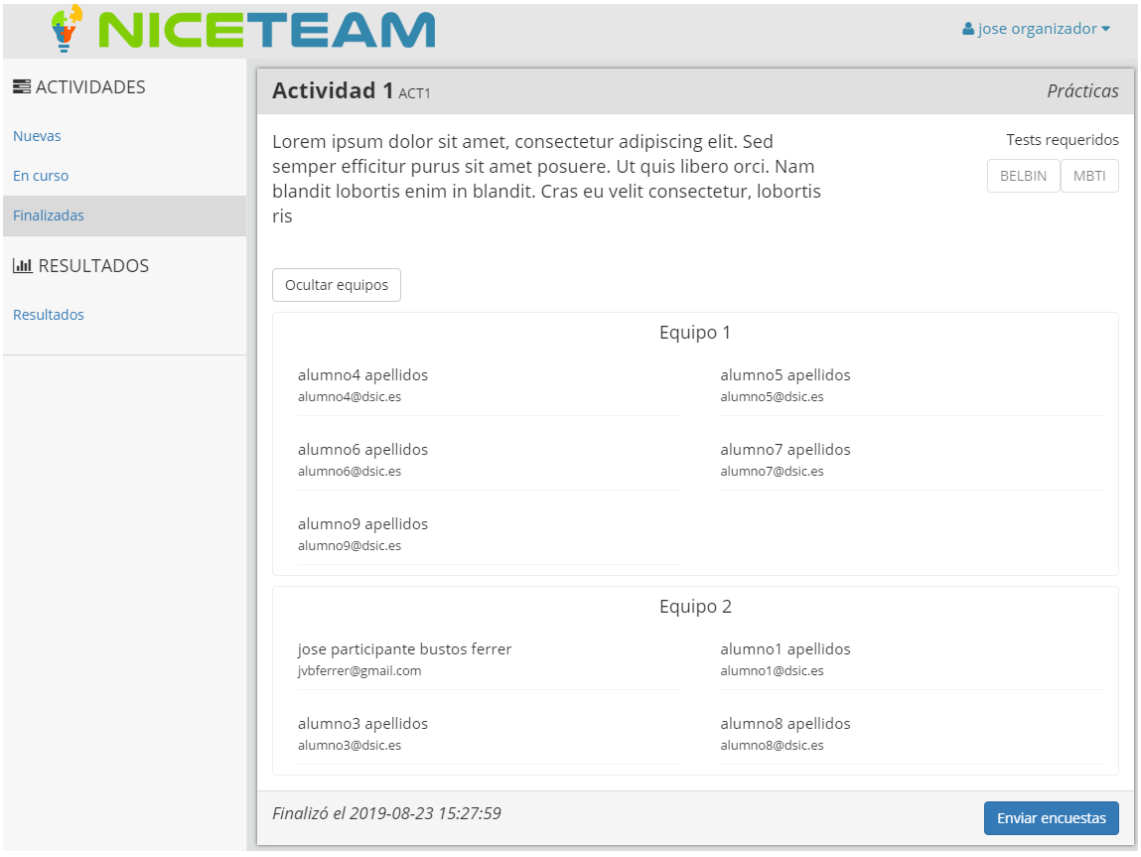
Actividad en curso, con posibilidad de finalizarla manualmente

Páginas de actividades finalizadas de un participante



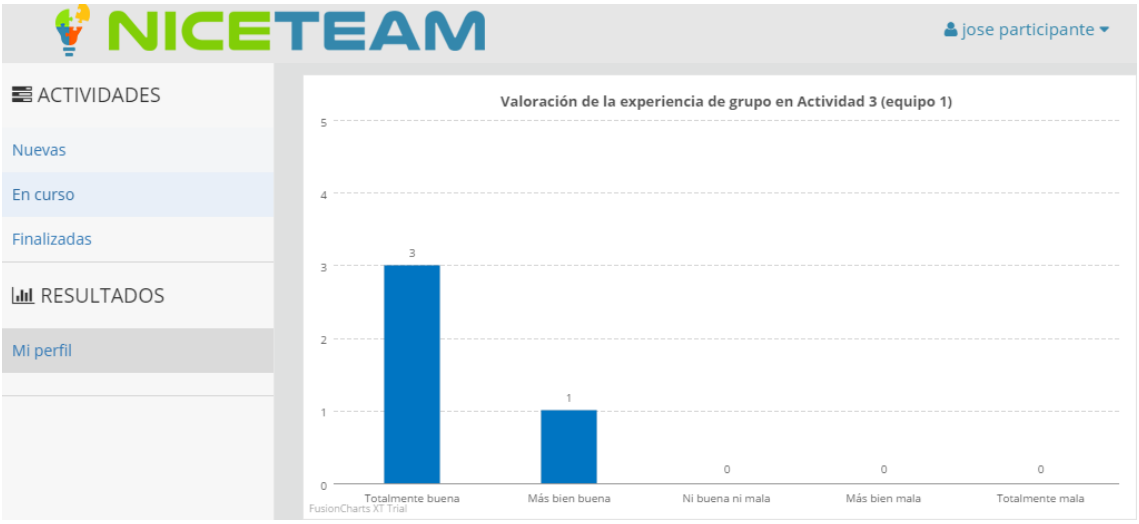
Historial de actividades finalizadas. Desde aquí se rellenará las encuestas

Páginas de actividades finalizadas de un organizador

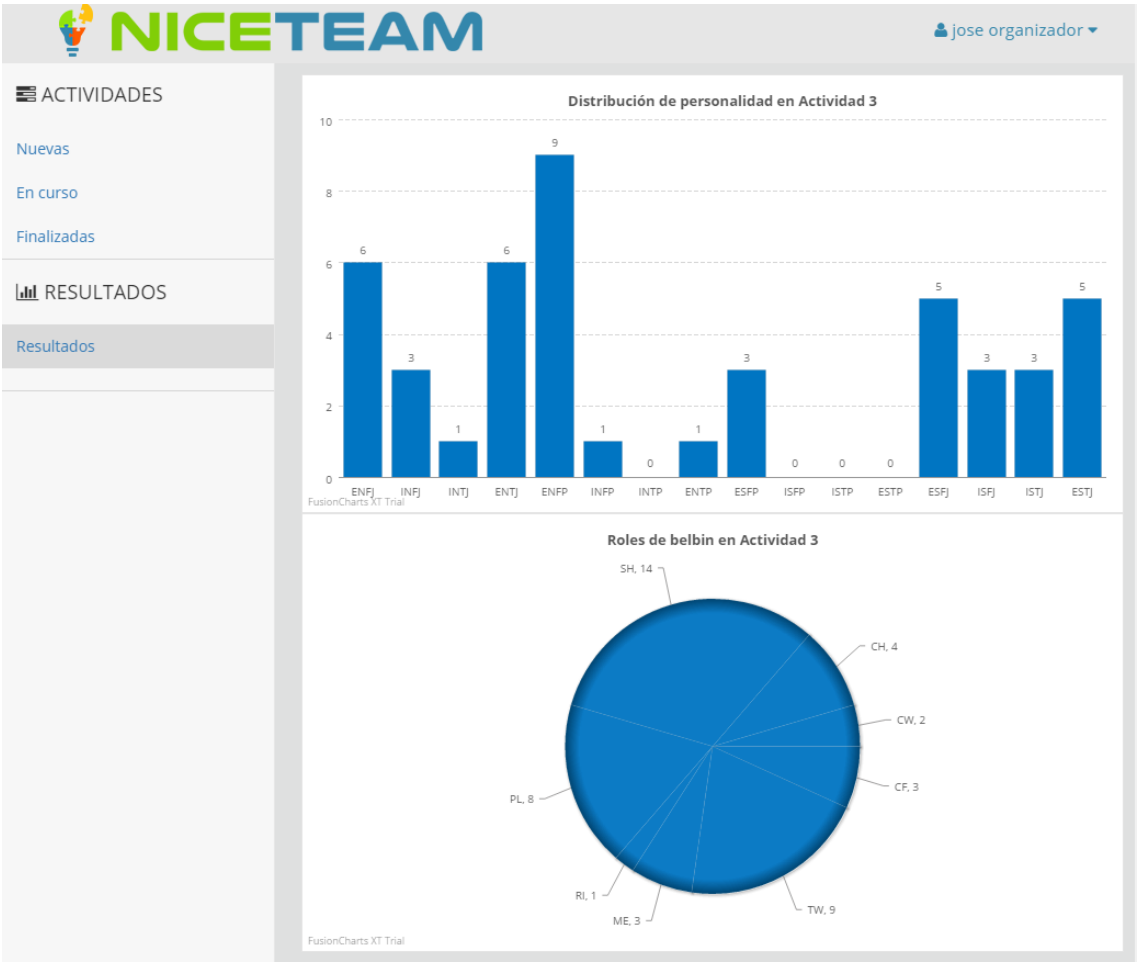


Historial de actividades finalizadas. Desde aquí se activa el envío de encuestas

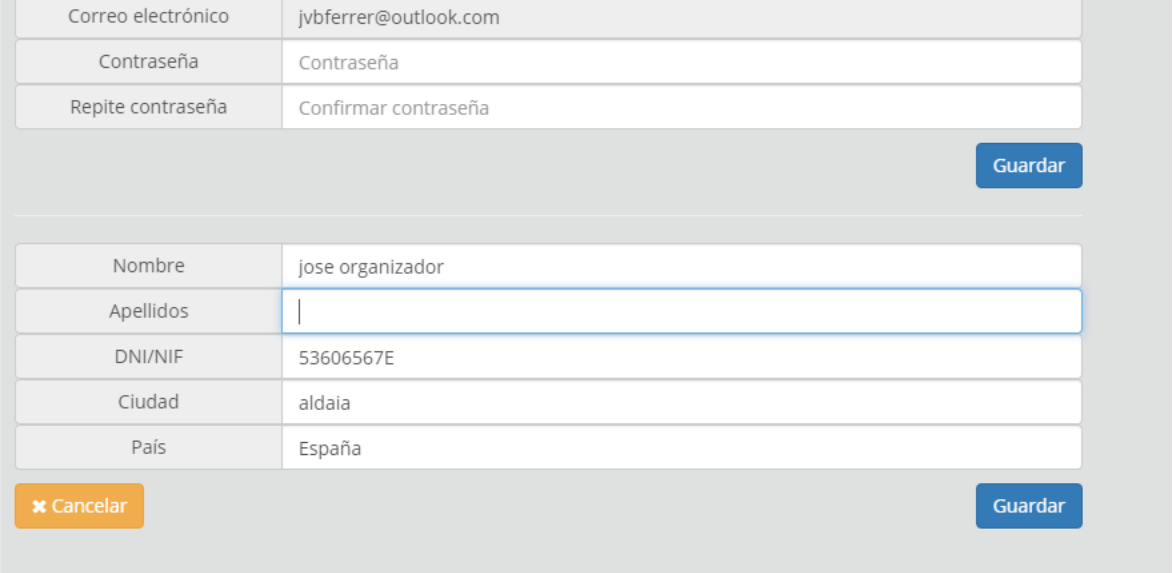
Página de resultados personales



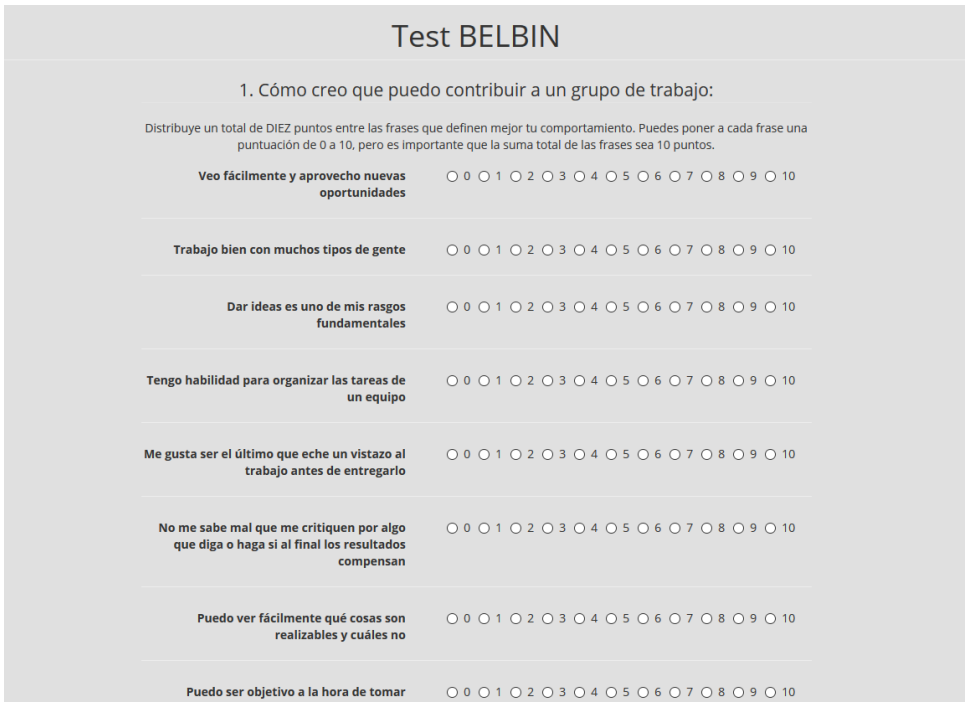
Página de resultados de equipo



Página de modificación de datos de cuenta



Página de test/encuesta



2.4 Pruebas de validación

Tal y como vimos en el proceso de *software*, es necesario probar y validar nuestro proyecto antes de darlo por finalizado.

La funcionalidad de los métodos y del flujo del programa ha sido validada mediante prueba manuales durante el desarrollo del proyecto. La aplicación es copiada al servidor *web* local en cada guardado del entorno de desarrollo y, a continuación, se realizan pruebas de inserción, modificación o alteración de datos con el fin de encontrar errores lógicos o sintácticos.

Estas pruebas incluyen tests unitarios en nuestro programa que especifican un escenario concreto con los diferentes tipos de usuario:

- Común
    - Iniciar sesión.
    - Registrarse con código de invitación.
    - Modificar información de la cuenta.
    - Cerrar sesión.
    - Cambiar de idioma.
- Organizador
    - Listar actividades.
    - Crear actividad.
    - Modificar actividad.
    - Iniciar una actividad.
    - Finalizar una actividad.
    - Enviar las encuestas de una actividad finalizada.
    - Invitar a participante.
    - Expulsar a participante.
    - Generar equipos.
    - Deshacer equipos.
    - Comprobar los resultados de las gráficas estadísticas (con datos de prueba).
- Participante
    - Listar actividades.
    - Rellenar un test.
    - Unirse a actividad.
    - Rellenar encuesta.
    - Mostrar gráficas con resultados personales.

En todos los casos, las pruebas se realizan probando primero la acción desarrollada independientemente desde el código y después realizándola mediante la interfaz desde el entorno de pruebas en un navegador web.

Tras añadir una acción nueva, se repite la prueba mediante interfaz de forma que interactúe con las otras acciones ya implementadas, hasta asegurarnos del correcto funcionamiento general.

Por otra parte, también es necesario validar nuestra interfaz basándonos en estándares de **usabilidad** y **accesibilidad,** como indicamos en el análisis del proyecto.

Respecto a la usabilidad, debemos tener en cuenta aspectos como [31]:

- **Objetivos** del sitio web: ¿están bien definidos? ¿Los contenidos ofrecidos se corresponden?
  Sí, gracias al análisis previo, los contenidos de la aplicación se ajustan a los objetivos planteados inicialmente.

- Los **contenidos**, ¿son mostrados de forma precisa y completa?
  Sí, separando correctamente las distintas áreas de acción de la página.

- ¿La **estructura** del sitio *web* está orientada al usuario?
  Sí, mediante dos menús contextuales en las partes superior e izquierda de la página, y con una zona principal en el centro.

- ¿Es **coherente** y **reconocible** el diseño general de la aplicación?
  El diseño y estructura están basados en la misma plantilla de *Bootstrap*.

- Respecto al **logo** del sitio, ¿es significativo y suficientemente identificable?
  Con letras grandes, un único icono identificativo y colores similares.

- ¿Se ofrece **información** sobre la empresa o el sitio *web*?
  En una página *Acerca de*.

- El sitio, ¿habla el mismo **idioma** que sus usuarios?
  Se ha implementado un sistema para elegir y mostrar los textos de la *web* en diferentes idiomas. El sistema es ampliable.

- ¿Se emplea un **lenguaje** claro, conciso y amigable?
  Los textos elegidos son de longitud moderada, enfocados a usuarios mayoritariamente jóvenes.

- Los **rótulos** y **enlaces**, ¿son significativos? ¿Y fácilmente reconocibles?
  Se han incluido los enlaces estrictamente necesarios, con rótulos únicos e identificables.

- ¿La estructura de **organización** y **navegación**, ¿es la más adecuada?
  Dada la naturaleza de la aplicación, se optó por utilizar un tema de *Bootstrap* adaptado a paneles de administración y aplicaciones *web*.

- ¿Es **predecible** la respuesta del sistema?
  Los botones y enlaces tienen un título suficientemente informativo.

- ¿Existen elementos que ayuden al usuario a **orientarse** en el sitio?
  Sí, mediante el menú lateral, que destaca la categoría en la que se encuentra el usuario.

- ¿Se evita la **redundancia** de enlaces?
  Se ha procurado no incluir más de un enlace que apunte al mismo sitio.

- ¿Hay páginas **huérfanas**?
  No.

- ¿Se evita la **sobrecarga informativa**?
  Sí, separando la información en secciones diferentes y procurando mantener suficiente separación entre los elementos visuales.

- La interfaz, ¿es clara y limpia, sin **ruido visual**?
  Los elementos visuales mostrados son los estrictamente necesarios.

- Las imágenes, ¿están bien **recortadas** y se ha cuidado su **resolución**?
  Se ha evitado la sobrecarga de imágenes, y las pocas utilizadas, se han diseñado específicamente para su uso particular.

- ¿Se informa al usuario de lo que está ocurriendo?
  En acciones no instantáneas, como la generación de equipos, se indica el estado en el que se encuentra (cargando).

- ¿Posee el usuario **libertad** para actuar?
  No se han incluido animaciones que no se puedan saltar, no se impide el uso del *click* derecho del ratón para obtener las propiedades de la página…

- ¿Se controla el **tiempo de respuesta**?
  Sí, se ha optimizado el código y evitado el uso de imágenes o archivos con tamaño demasiado grande.

En cuanto a la accesibilidad, también tendremos en cuenta:

- El **tamaño de la fuente** y la **tipografía,** ¿facilita la legibilidad del texto?
  Se ha cuidado la tipografía utilizada, eligiendo *Open Sans*, la cual destaca por su fácil legibilidad. El tamaño de la fuente se adapta al uso que se le quiera dar, aunque se han utilizado tamaños relativos más grandes.

- ¿Existe un **alto contraste** entre el color del texto y del fondo?
  Sí.

- Las imágenes, ¿incluyen el **atributo *alt*** que describe su contenido?
  Sí.

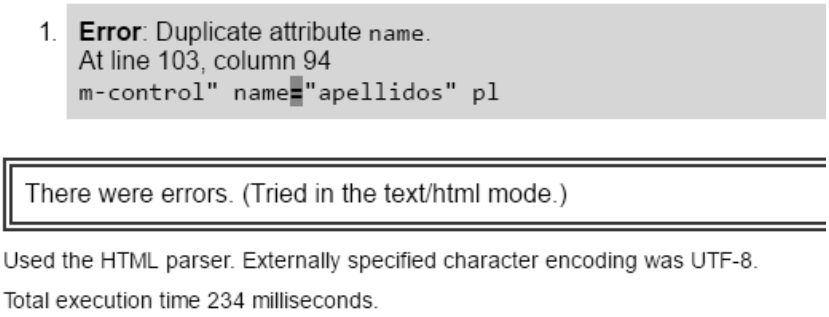
- ¿Es compatible el sitio con **diferentes navegadores**?
  Sí, se ha probado con *Google Chrome*, *Mozilla Firefox*, *Internet Explorer*, *Microsoft Edge* y *Opera Browser*.

- ¿Puede el usuario usar el sitio **sin descargar e instalar extensiones** adicionales?
  Sí.

Además, existen diferentes herramientas *online* que permiten medir la accesibilidad de nuestro sitio [32]. Para ello, ha sido necesario permitir el acceso libre desde Internet al servidor *web* local de pruebas, mediante una redirección de puertos a través del *router* personal.

- Validador de *HTML5*: https://html5.validator.nu

Esta herramienta nos permite identificar errores o incompatibilidades de etiquetado o estructura respecto al estándar de *HTML 5*.



1. **Error**: Duplicate attribute `name`.
   At line 103, column 94
   `m-control" name="apellidos" pl`

There were errors. (Tried in the text/html mode.)

Used the HTML parser. Externally specified character encoding was UTF-8.

Total execution time 234 milliseconds.

Aparte de ese único error leve sobre el cual se toman las medidas oportunas eliminando atributos duplicados en algunos elementos de la página, no se han encontrado más problemas de etiquetado *HTML*.

- Validador *CSS3* del *World Wide Web Consortium* (*W3C*): http://jigsaw.w3.org/css-validator/

Resultados del Validador CSS del W3C para [redacted] (CSS versión 3)

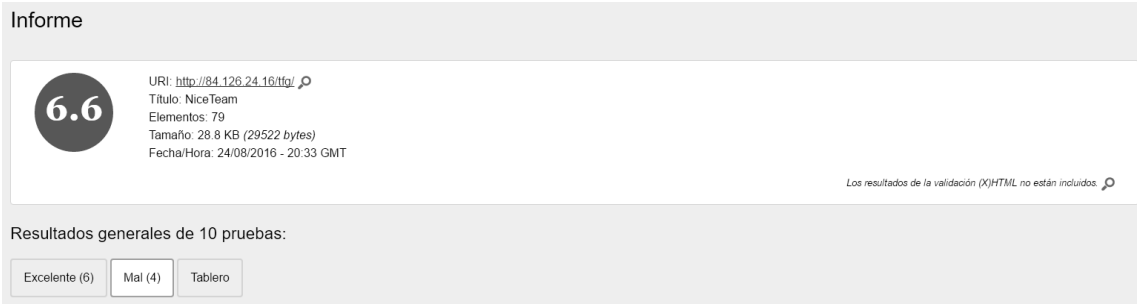
Disculpas! Hemos encontrado las siguientes errores (32)

URI : http://[redacted]ttfg/view/css/bootstrap.min.css

5	input[type="checkbox"], input[type="radio"]	Propiedad no válida : margin-top dimensión desconocida 1px\9
5		La propiedad -webkit-min-device-pixel-ratio no existe en el medio screen){input[type=dat height:34px}.input-group-sm input[type=date],.input-group-sm input[type=time],.input-grou sm,input[type=datetime-local].input-sm,input[type=month].input-sm{line-height:30px}.inpu input[type=month],input[type=date].input-lg,input[type=time].input-lg,input[type=datetime-
5	.checkbox input[type="checkbox"], .checkbox-inline input[type="checkbox"], .radio input[type="radio"], .radio-inline input[type="radio"]	Propiedad no válida : margin-top dimensión desconocida 4px\9
5	.form-control-feedback	La propiedad pointer-events no existe : none
5	.btn	La propiedad touch-action no existe : manipulation
5	.btn	La propiedad user-select no existe : none
5	a.btn.disabled, fieldset[disabled] a.btn	La propiedad pointer-events no existe : none
5	.caret	Propiedad no válida : border-top solid\000009 no es un valor de color : 4px solid\000009
5	.dropdown-menu > .disabled > a:focus, .dropdown-menu > .disabled > a:hover	Tentativa de encontrar un punto y coma antes del nombre de la propiedad. Añádalo
5	.dropdown-menu > .disabled > a:focus, .dropdown-menu > .disabled > a:hover	La propiedad progid no existe : DXImageTransform
	.dropdown-menu >	

De entre los errores leves encontrados, todos dependen de las librerías utilizadas en el proyecto, como se indica en la *URI* del archivo con incidencias (*bootstrap.min.css*).

- Evaluador de accesibilidad web mediante Pautas de Accesibilidad para el Contenido Web 2.0 (WCAG 2.0): http://examinator.ws/



Aunque la aplicación *web* aprueba la evaluación, realizamos las acciones correctivas necesarias para asegurarse de que se cumple totalmente con los estándares. Algunas de estas acciones incluyen:

- Asegurarse de incluir en todas las imágenes textos alternativos.
- Además de incluir constantes con el texto de la página en diferentes idiomas, hay que cerciorarse de indicar en el etiquetado *HTML* el idioma en uso.

- Validador de accesibilidad *web* WCAG 2.0 con nivel de conformidad **AA**: http://www.tawdis.net/index.html?lang=es

De la misma forma, para este nivel de conformidad, nuestra aplicación consigue superar la mayoría de pruebas, requiriendo únicamente un mínimo ajuste para conseguir la excelencia en estándares:

- Textos alternativos a imágenes y otros elementos.
- Establecer títulos para cada página.
- Incluir nombre y rol en componentes de la interfaz.

A pesar de las advertencias mostradas y aunque la validación de nuestro sitio es satisfactoria incluso sin haber desarrollado el producto en su completa extensión, es necesario repasar las sugerencias, advertencias y errores para garantizar una experiencia de uso excelente.

Tras tomar las acciones correctivas individualmente para cada observación, tenemos un producto que cumple con los mínimos estándares de accesibilidad y usabilidad y podrá ser usado por múltiples usuarios con circunstancias diferentes.

Resumiendo:

Primero hemos desarrollado nuestra solución basándonos en plataformas estándares que permiten una rápida integración de los requerimientos de usabilidad y accesibilidad.

Después, hemos recurrido a la teoría de estándares, siguiendo consejos e indicaciones de diferentes fuentes y organizaciones, y hemos sometido a nuestro programa a pruebas objetivas para validar su estructura e implementación.

Por último, analizamos las medidas correctivas sugeridas y las aplicamos para alcanzar el nivel de conformidad requerido.

3 Conclusiones

3.1 Trabajo realizado

Para realizar nuestro proyecto primero hemos necesitado recopilar toda la información necesaria mediante entrevistas y reuniones, pero después hemos necesitado desarrollar un plan de negocio:

- Investigamos sobre el contexto de la problemática y las soluciones similares actuales.
- Repasamos las tecnologías necesarias para realizar la implementación y se realiza el estudio de los requerimientos del programa, modelando una solución mediante el Lenguaje Unificado de Marcas (*U.ML.*, por sus siglas en inglés).
- Después elegimos la arquitectura de la aplicación y el proceso de *software* para mantener una organización durante el desarrollo.
- Realizamos bocetos más específicos de la solución planteada para encontrar posibles fallos de concepto o estructura y, tras validar un resultado, la implementamos.
- Una vez implementada, validamos el resultado con los requerimientos obtenidos en el análisis previo, hasta que sea satisfactorio.
- Validamos la lógica, estructura, accesibilidad y usabilidad de nuestro programa mediante códigos propios, herramientas *online* y los principios generales más teóricos de la usabilidad.

Por ello, en cuanto a los objetivos planteados inicialmente y especificados en el análisis del proyecto, podemos afirmar que los hemos alcanzado.

Se ha desarrollado una aplicación administrable en la que podemos dar solución a la problemática inicial: necesidad de gestionar actividades en grupo y formar los equipos a través de un algoritmo externo ya desarrollado. Es posible generar y administrar equipos de trabajo, realizar un seguimiento de las actividades y del éxito de sus grupos en cuanto al objetivo de la tarea mediante resultados estadísticos obtenidos con encuestas de satisfacción.

Además, ahora tenemos control sobre una aplicación que puede ser ampliada, mejorada, optimizada y administrada independientemente de otras soluciones educativas, como PoliformaT.

3.2 Estudio para su uso en otros entornos no académicos

En el estado actual, no se ha implementado el perfil de Administrador debido a su redundancia en el caso de un entorno académico y controlado. En cualquier caso, gracias al patrón arquitectónico Modelo-Vista-Controlador implementado, es posible crear nuevos perfiles de usuarios e integrarlos con relativa sencillez.

De igual forma, gracias al sistema de idiomas usado, podemos crear, modificar o eliminar tantos idiomas como se necesite en la aplicación.

También sería interesante desarrollar en un futuro algún sistema de *chat* o contacto entre Participantes y Organizadores, de forma que la comunicación sea directa y constante incluso si los integrantes de la Actividad se encuentran en lugares físicamente distintos.

3.3 Valoración personal

Personalmente, me ha gustado desarrollar este trabajo, dado que pertenece a la rama tecnológica a la que me dedico. He podido investigar y aprender más sobre tecnologías ampliamente usadas en el mundo laboral y que abren la puerta a desarrollos más complejos.

Tampoco ha resultado demasiado difícil ya que muchas tecnologías las hemos visto en asignaturas del Grado en Ingeniería Informática, lo cual ayudaba bastante al planificar el desarrollo y los métodos para implementarlo. Quizás el uso de librerías o herramientas más complejas ayuden en el desarrollo a medio plazo, pero no permiten conocer las bases en las que se fundamenta un programa basado en la nube.

Sin duda, repetiría.

Anexo

http://stackoverflow.com/questions/6782686/return-false-success-in-ajax-jquery

https://api.jquery.com

http://bootsnipp.com/snippets/featured/login-and-register-tabbed-form

http://stackoverflow.com/questions/2019608/pass-entire-form-as-data-in-jquery-ajax-function

http://stackoverflow.com/questions/1792603/how-do-i-php-unserialize-a-jquery-serialized-form

http://stackoverflow.com/questions/503093/how-can-i-make-a-page-redirect-using-jquery

http://stackoverflow.com/questions/5404839/how-can-i-refresh-a-page-with-jquery

http://php.net/manual/es/filter.filters.sanitize.php

https://jqueryvalidation.org/validate/

http://techtastico.com/post/cual-es-el-tamano-maximo-de-una-direccion-de-email/

http://stackoverflow.com/questions/1846202/php-how-to-generate-a-random-unique-alphanumeric-string

http://stackoverflow.com/questions/14922208/can-i-use-varchar32-for-md5-values

http://stackoverflow.com/questions/2475065/what-data-type-to-use-in-mysql-to-store-images

http://www.fusioncharts.com/dev/using-with-server-side-languages/php/creating-charts-with-data-from-a-database.html

Bibliografía

[1] **WIKIPEDIA**, HISTORIA DE INTERNET. ÚLTIMA MODIFICACIÓN EL 18 MAYO 2016 A LAS 21:58.
https://es.wikipedia.org/wiki/Historia_de_Internet

[2] **WIKIPEDIA**, ANEXO: PAÍSES POR NÚMERO DE USUARIOS DE INTERNET. ÚLTIMA MODIFICACIÓN EL 23 MAYO 2016 A LAS 21:50.
https://es.wikipedia.org/wiki/Anexo:Países_por_número_de_usuarios_de_Internet

[3] **WORLD WIDE WEB CONSORTIUM (W3C) ESPAÑA**, GUÍA BREVE SOBRE ESTÁNDARES WEB. ÚLTIMA CONSULTA EL 31 MAYO 2016 A LAS 18:35.
http://www.w3c.es/Divulgacion/GuiasBreves/Estandares

[4] **SAKAI PROJECT**, ABOUT SAKAI.
https://www.sakaiproject.org/

[5] **DITRENDIA**, INFORME MOBILE EN ESPAÑA Y EN EL MUNDO 2015. PÁGINA 5, USO DEL MÓVIL.
http://www.ditrendia.es/wp-content/uploads/2015/07/Ditrendia-Informe-Mobile-en-España-y-en-el-Mundo-2015.pdf

[6] **WORLD WIDE WEB CONSORTIUM (W3C)**
https://www.w3.org/TR/html5/

[7] **WIKIPEDIA**, TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB, POR **JOSÉ MONTOYA GUZMÁN**.
https://en.wikipedia.org/wiki/Web_application_development

[8] **PREZI.COM**, TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB, POR **JOSÉ MONTOYA GUZMÁN**.
https://prezi.com/ohanpxoyzgyx/13-tecnologias-para-el-desarrollo-de-aplicaciones-web/

[9] **WIKIPEDIA**, DOCUMENT OBJECT MODEL. ÚLTIMA MODIFICACIÓN EL 13 MAYO 2016 A LAS 18:25.
https://es.wikipedia.org/wiki/Document_Object_Model

[10] **JQUERY**, API DOCUMENTATION
http://api.jquery.com/jquery.ajax/

[11] **LIBROSWEB.ES**, INTRODUCCIÓN A AJAX: CAPITULO 1
https://librosweb.es/libro/ajax/capitulo_1.html

[12] **NICEARMA.COM**, JAVA VS PHP. 8 DE SEPTIEMBRE, 2015
http://www.nicearma.com/2015/09/08/java-vs-php/

[13] **JAVA,** ¿QUÉ ES JAVA?
https://www.java.com/es/about/whatis_java.jsp
https://www.java.com/es/download/faq/whatis_java.xml

[14] **WIKIPEDIA**, JAVA (LENGUAJE DE PROGRAMACIÓN). ÚLTIMA MODIFICACIÓN EL 2 DE JUNIO 2016 A LAS 12:29
https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)

[15] http://getbootstrap.com/getting-started/, http://getbootstrap.com/css/, http://getbootstrap.com/components/, http://getbootstrap.com/javascript/

[16] **DIGITALOCEAN.COM**, SQLITE VS MYSQL VS POSTGRESQL: A COMPARISON OF RELATIONAL DATABASE MANAGEMENT SYSTEMS, POR **O.S. TEZER**. 21 DE FEBRERO 2014.
https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[17] **WIKIPEDIA**, ANEXO: COMPARACIÓN DE SISTEMAS ADMINISTRADORES DE BASES DE DATOS RELACIONALES. ÚLTIMA MODIFICACIÓN EL 29 DE SEPTIEMBRE DE 2015 A LAS 14:02.
https://es.wikipedia.org/wiki/Anexo:Comparación_de_sistemas_administradores_de_bases_de_datos_relacionales

[18] **YISES.COM**, DIFERENCIAS ENTRE MYSQL Y SQLITE: ¿CUÁNDO USAR CADA UNO?, POR JESÚS E. PÉREZ VILLEGAS. 15 DE ABRIL DE 2014
http://yises.com/blog/2014/04/15/diferencias-entre-mysql-y-sqlite-cuando-usar-cada-uno/

[19] **PHP.NET**, HISTORIA DE PHP.
http://php.net/manual/es/history.php.php

[20] https://www.visual-paradigm.com/download/community.jsp

[21] **INGENIERÍA DEL SOFTWARE, GRADO EN ING. INFORMÁTICA, U.P.V.** *Tema 2: El proceso de* software *y Tema 3: Modelado O.O. con U.M.L.*

[22] http://periodico.laciudadaccesible.com/portada/opinion-la-ciudad-accesible/item/866-que-es-la-usabilidad, https://www.upf.edu/hipertextnet/numero-2/diseno_web.html, http://www.adrenalina.es/mejorar-usabilidad-web/, http://www.usabilitynet.org/tools/r_international.htm

[23] https://balsamiq.com/products/mockups/

[24] **CÓMO ELEGIR ENTORNO DE DESARROLLO WEB**, POR LUIS DEL VALLE HERNÁNDEZ. 20 DE JULIO DE 2015
http://programarfacil.com/podcast/36-como-elegir-el-entorno-de-desarrollo-web/

[25] https://es.wikipedia.org/wiki/Eclipse
http://www.nicolasbize.com/blog/the-best-ide-in-the-world/

[26] https://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[27] http://soyprogramador.liz.mx/appserv-wamp-o-xamp/

[28] http://www.fusioncharts.com/dev/using-with-server-side-languages/php/creating-charts-with-data-from-a-database.html

[29] http://fontawesome.io/icons/, http://glyphicons.com/

[30] https://startbootstrap.com/template-overviews/sb-admin-2/

[31] https://www.nngroup.com/articles/ten-usability-heuristics/,
http://www.nosolousabilidad.com/articulos/heuristica.htm

[32] http://labda.inf.uc3m.es/awa/en/node/125, http://examinator.ws/,
http://www.tawdis.net/, http://webaccesible.cea.es/?q=automaticas