**Proyecto Final**

Universidad Tecnológica Nacional

Facultad Regional Mendoza

Ingeniería en Sistemas de Información



**Sale Partido**

Año 2016

Integrantes: Cuerpo docente:

Álvarez, Martín – Legajo: 34849

García, Mauricio – Legajo: 34889

Gómez, Federico – Legajo: 34497

Suarez, Adriel – Legajo: 34480

Profesor Titular: Vázquez, Alejandro

Profesor Adjunto: Moralejo, Raúl

JTP: Manino, Gustavo

Ayudante de 1ra: Villa, Diego

**TRABAJO PRÁCTICO N°1: Desarrollo de un Sistema de Información Real**

Resumen Técnico

Justificación del trabajo a realizar

En la actualidad en Mendoza ha ocurrido una gran expansión en cuanto a la construcción de canchas de fútbol de pasto sintético que ha llevado a que mucha gente alquile canchas para jugar con amigos, participar en torneos, abrir escuelitas de fútbol infantiles, dándole gran utilidad a dichos espacios para la recreación. Cada vez son más frecuentes en las redes sociales como en servicios de mensajería instantánea para teléfonos inteligentes los grupos con amigos y amigos de amigos para organizar un partido de fútbol, y en algunos casos una persona puede estar en varios grupos distintos a la vez.

Objetivos

Mediante el desarrollo del proyecto “**sale partido**” se pretende aportar una herramienta integral tanto para jugadores como para el propietario de las canchas de manera que el proyecto permita facilitar la organización del evento deportivo para distintos interesados y realización de la reserva correspondiente al lugar seleccionado, de esta manera facilitaría la gestión de turnos para los propietarios. Disminuyendo así el tiempo gastado por los organizadores del evento, las llamadas telefónicas realizadas por un lado y la cancelación de reservas a último momento que generan pérdidas a los propietarios de las canchas por el otro.

Conclusiones

Relevamiento general

Relevamiento de las funcionalidades

Para poder conocer al entorno y saber qué es lo que realmente se necesita, investigamos sistemas que tengan funcionamientos similares o bien, algún módulo que tenga un comportamiento y que pueda llegar a ser útil a la hora de implementar el sistema.

FULBO



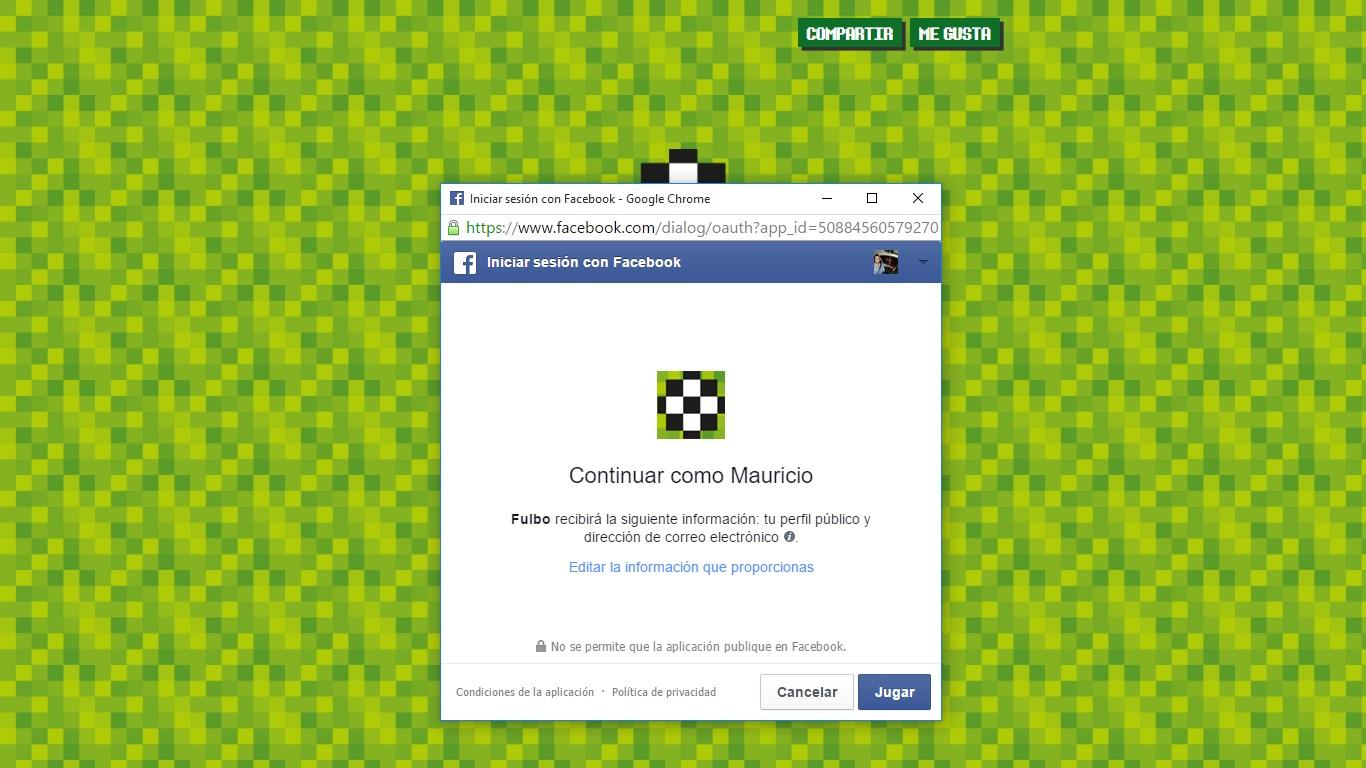
Link: <http://www.jugafulbo.com.ar>

Descripción.

Es un sitio web orientado a la organización de partidos de fútbol.

Características.

El propósito del sitio web es organizar partidos de fútbol. Para usar dicho sitio web, el interesado primero debe registrarse utilizando una cuenta de Facebook. Una vez que el usuario se registra y realiza su correspondiente Login, tiene acceso al módulo de partido, módulo de canchas y módulo de crear partido.



Una vez que se realiza el registro, nos encontramos con una pantalla principal, donde se pueden observar los módulos con los que cuenta la aplicación y una ventana principal que detalla los partidos organizados por el usuario o a los que ha sido invitado, pero que todavía, no se han jugado. También se puede ver una segunda ventana, que muestra el historial de partidos jugados.





Funciones destacadas e interfaces.

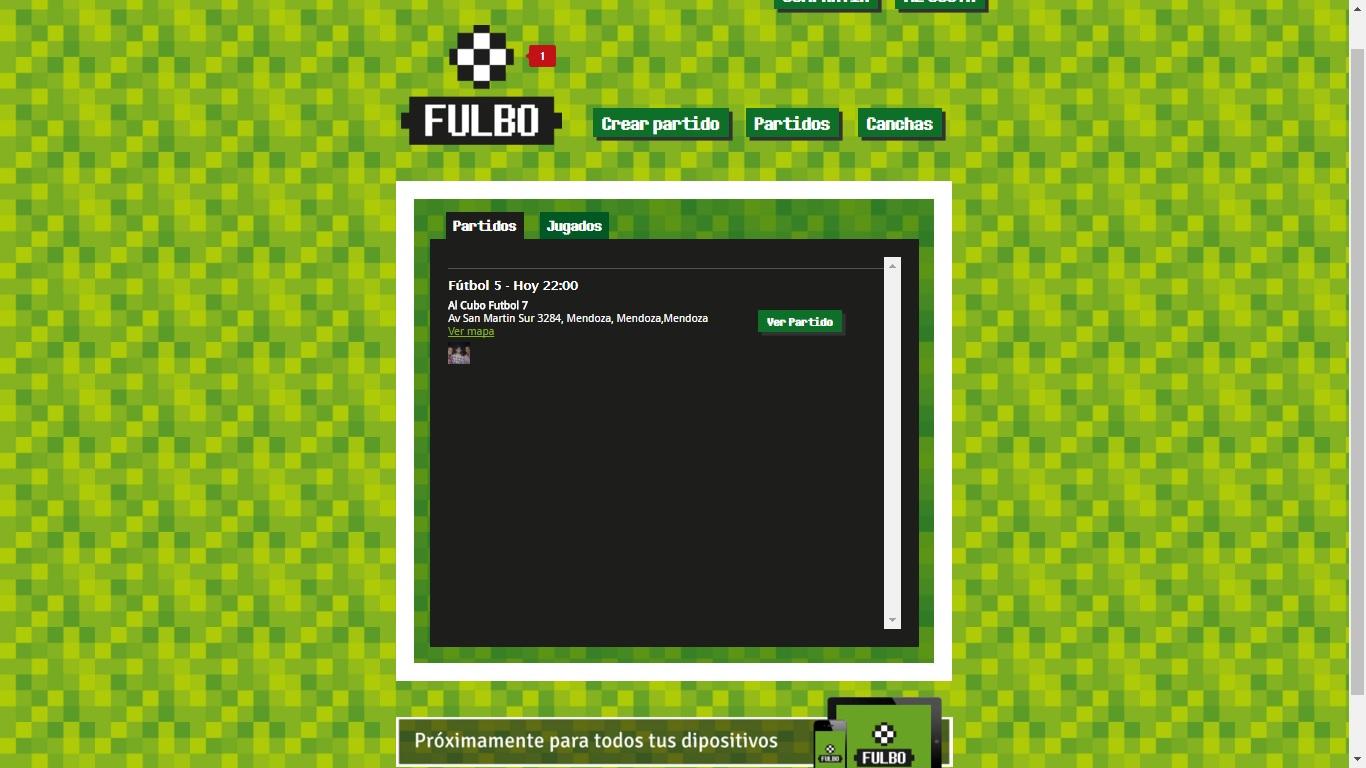
**Crear partido**

En el módulo de crear partido, se puede observar campos de texto que deben ser llenados con sus respectivos datos.



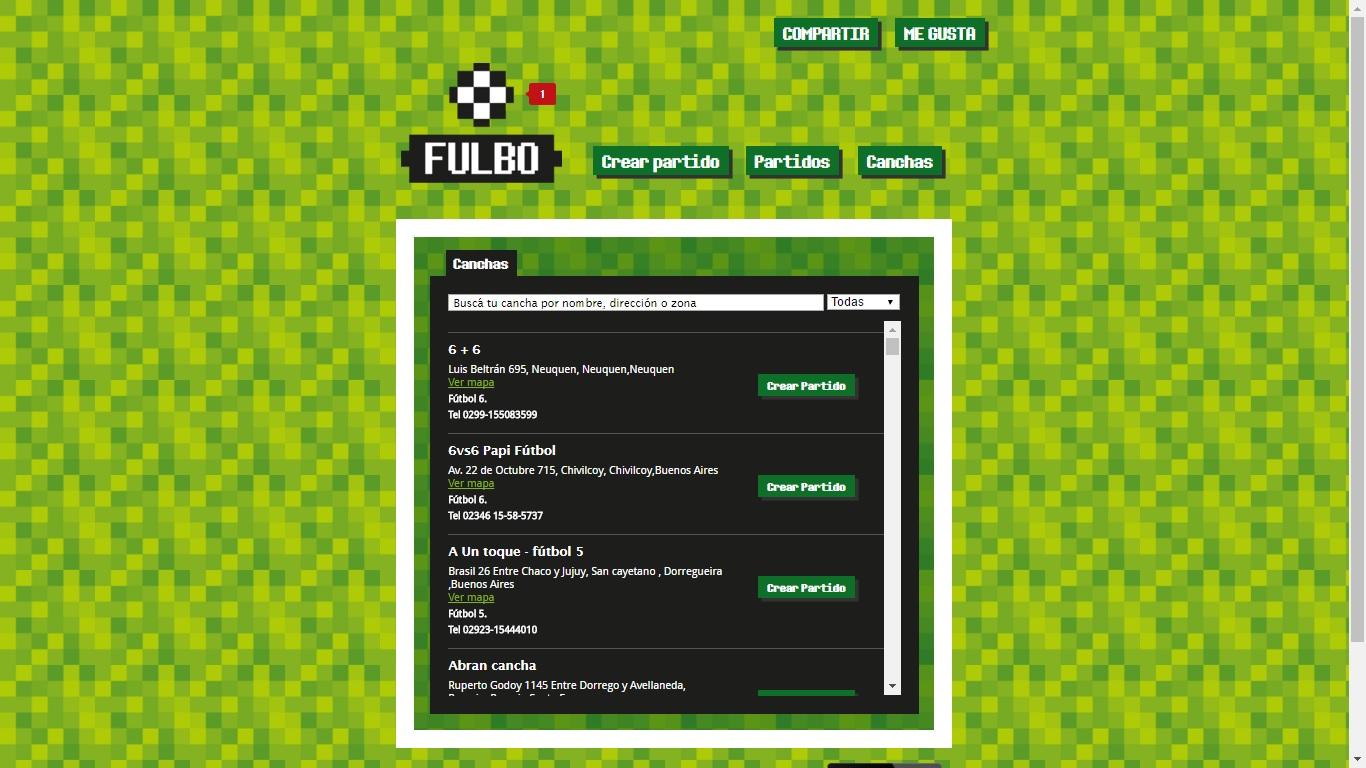
**Partidos**

En el módulo partidos, solo se puede observar la misma ventana que se observa en el inicio de la página web, luego de haber hecho el Login correspondiente.



**Canchas**

El módulo canchas muestra todas las canchas asociadas al sitio, donde se pueden llevar a cabo los encuentros organizados.



FUBLES



Link: <https://es.fubles.com/>

Descripción

Fubles, la red social que te permite de organizar partidos de fútbol, baloncesto, voleibol y de casi todos los deportes.

Características

Permite inscribirse a partidos programados en la zona donde se encuentra u organizar uno mismo el partido.   
Una vez que se elige el lugar del partido, se pone el partido en el sistema y Fubles envía una convocatoria a todos los jugadores disponibles a tu alrededor.   
Si solo se quiere jugar un partido con los amigos, se crea un juego privado y Fubles comunica el evento solo a tu gente.

Funcionalidades destacadas

**Registrarse en la aplicación**

Permite registrarse a través de Facebook o creando una cuenta en Fubles ingresando un correo valido.

Al crear una cuenta solicita:

* Correo. (String)
* Contraseña. (String)
* Nuestra ubicación (Zona) encontrada usando el GPS del móvil.

El correo debe confirmarse, ingresando el código que la aplicación envía al mail.

|  |  |
| --- | --- |
| C:\Users\tchgc\Desktop\Fubles\Screenshot_2016-03-22-15-21-03.png | C:\Users\tchgc\Desktop\Fubles\Screenshot_2016-03-22-15-27-18.png |
| C:\Users\tchgc\Desktop\Fubles\Registro01.png |  |

**Búsqueda de Jugadores**

La pantalla principal de la aplicación cuenta con una barra de búsqueda, la cual permite encontrar jugadores registrados una vez encontrado se puede ver todo su perfil, mandar un mensaje o seguir.

|  |  |
| --- | --- |
| C:\Users\tchgc\Desktop\Fubles\PantallaPrincipal.png | C:\Users\tchgc\Desktop\Proyecto\Fubles\buscarJugadores.png |

**Establecer zona geográfica del usuario**

El usuario puede cambiar su zona geográfica para encontrar otros partidos.

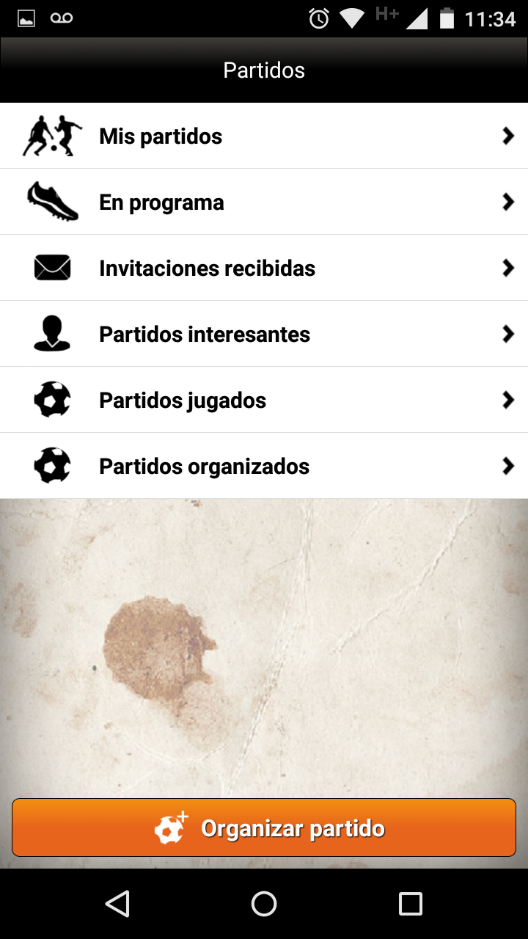
Esto lo puede hacer, escribiendo la dirección o buscándola en el mapa de Google Maps.



**Pestaña de Partidos**

Fubles comunica cuando hay partidos disponibles en tu entorno o cuando un amigo te invita a jugar. Selecciona el partido y ocupa uno de los roles disponibles en el equipo blanco o negro.

Aquí aparecerán los próximos partidos en los que está confirmado, los partidos ya jugados, los partidos invitados, los partidos próximos cerca de mi ubicación.



**Perfil del jugador**

En el perfil del jugador se puede ver:

Pestaña de Detalles.

* Su foto. (.jpg)
* Cantidad de partidos jugados. (int)
* Cantidad de partidos que confirmo y no concurrió. (int)
* Zona a la que pertenece.
* Nivel. (int)
* Rango. (int)

Pestaña de Opiniones.

* Comentarios de otros jugadores. (String)

Pestaña de partidos.

* Todos los partidos jugados. (int)



**Organizar partido**

Si es el primer partido que organiza se pide que ingrese un número de teléfono.

Luego busca automáticamente cancha cerca de tu zona. Si no está la cancha que se buscaba permite agregar nuevas canchas.

Una vez encontrado el lugar, muestra un mapa de la ubicación. Y perime agregar información sobre la cancha (nº jugadores, terreno, deporte, cubierto, campo).

Después se debe seleccionar:

1. Día y Hora.(Date)
2. Formas de pago. (String)
3. Cantidad de suplentes. (int)
4. Excluir debutantes o usuarios que no fueron a jugar en partidos pasados y que confirmaron.
5. Reservar puesto arquero.
6. Compartir en Facebook.
7. Privacidad. (boolean)

Para finalizar muestra una pantalla con los datos del partido aquí se puede:

* Cancelar el partido.
* Invitar a jugadores.
* Ver los jugadores que jugaran el partido.

|  |  |
| --- | --- |
| C:\Users\tchgc\Desktop\Fubles\DondeOrganizarPartido00.png | C:\Users\tchgc\Desktop\Fubles\DondeOrgPartido00.png |
| C:\Users\tchgc\Desktop\Fubles\DondeOrgPartido01.png | C:\Users\tchgc\Desktop\Fubles\Zona00.png |

CALECIO



Link: <http://www.calecio.com/>

Descripción

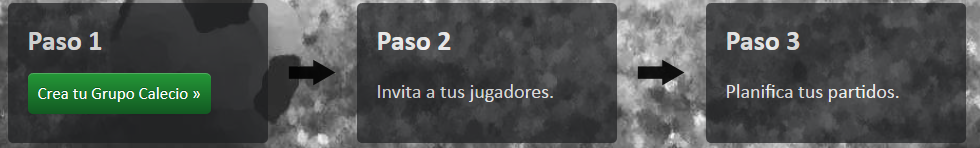
Calecio es un sitio web donde puedes armar grupos de deporte y crear partidos, es un sitio totalmente gratuito. Tiene como propósito la organización de partidos ya sea de futbol, básquet, vóley y tenis. También permite llevar un control de los partidos creados, visualizarlos en el calendario, mandar mensajes y otras funcionalidades.

Cuenta con una aplicación en Google Play y permite logearnos a través de Facebook o utilizando un perfil de Calecio.

Características

Calecio busca simplificar la forma de organizar los partidos mediante tres simples pasos:

1) Crear tu Grupo Calecio.  
2) Invita tus jugadores al grupo.  
3) Crea los partidos.

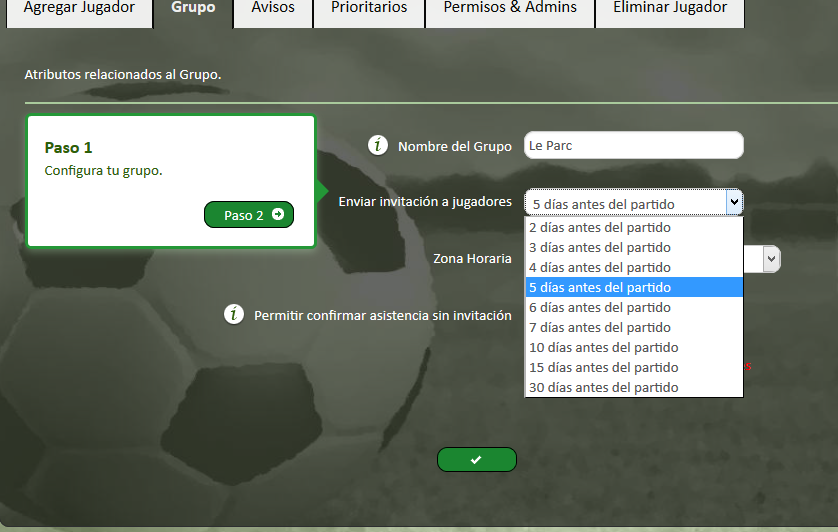


- Calecio enviará notificaciones para invitar, cancelar, o informar de cualquier cambio a cada jugador.  
- Confirma tu asistencia directamente desde la notificación, sin necesidad de entrar a la web.  
- Tu partido será automáticamente añadido al calendario en tu teléfono.  
- Todos los jugadores pueden seguir en todo momento el estado del partido y las posiciones en el campo.

Funcionalidades destacadas

**Invitación a los jugadores**

Permite enviar notificaciones sobre los partidos a crear, con un margen de tiempo (“x” días antes del partido) ajustable a la zona horaria.



**Pestaña de avisos**

Cuenta con una pestaña de avisos donde se puede marcar “si” o “no” según nuestras preferencias, ya sea para permitir bajas de jugadores en partidos confirmados, si queremos confirmar el partido o cancelarlo, mandar recordatorios y unas opciones más. Luego de seleccionar las mismas validamos y manda los emails en caso de ser necesario.



**Permisos de administrador**

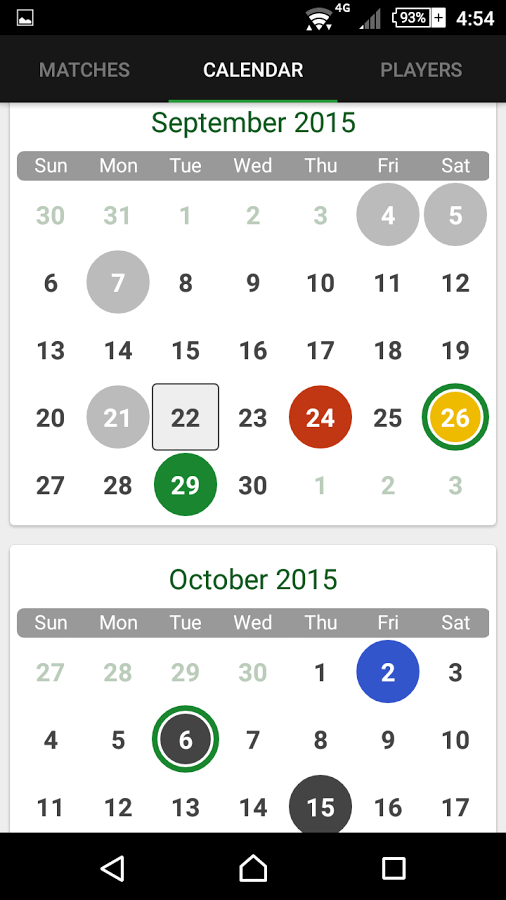
Se pueden otorgar permisos de administrador en el grupo, puede existir más de un administrador y le permite a los mismos eliminar jugadores, mandar avisos y gestionar el equipo. No se puede eliminar al administrador original es decir al creador del grupo. En la interface se muestra una ventana con los administradores.

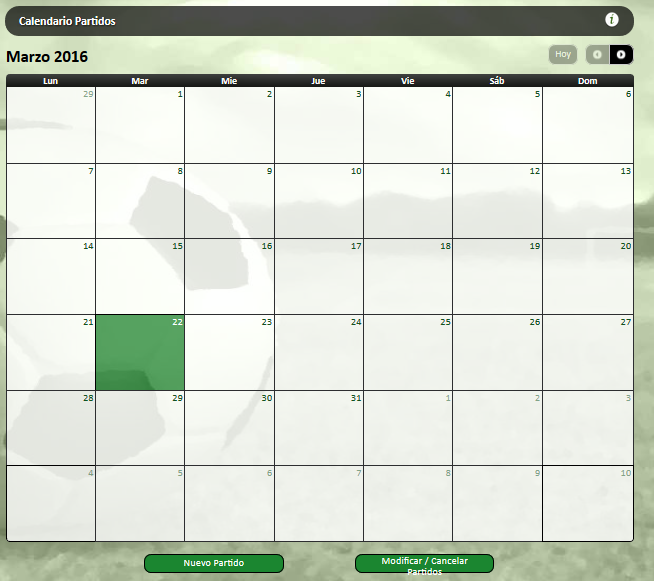


**Calendario**

El calendario está sincronizado con el celular, la aplicación gestiona los partidos y nos manda una notificación para el día y fecha del mismo. Se pueden crear nuevos partidos o modificar los mismos desde la interfaz del calendario.

Muestra con un contorno verde los partidos confirmados, y de distintos colores los partidos por tipo de deporte. Con gris sombreado nos indica que esas fechas y/o partidos ya se realizaron.





**Soporte de idiomas**

Cuenta con soporte para los siguientes idiomas: inglés, español, catalán, alemán, italiano, francés y portugués. Se puede apreciar en la parte superior de la página web.



DoLeague



Link: <http://app.doleague.com/es>

Descripción:

Es un sitio orientado a la gestión y organización de campeonatos y partidos de cualquier deporte. Posee una versión Free y una Premium.

Para acceder debemos crear una cuenta con nuestra dirección de e-mail y que servirá siempre para nuestro acceso a la web.

Características

DoLeague brinda la posibilidad de:

-Organizar campeonatos y partidos.

-Gestionar inscripciones y pagos a través de tarjetas de crédito.

-Gestionar sponsors.

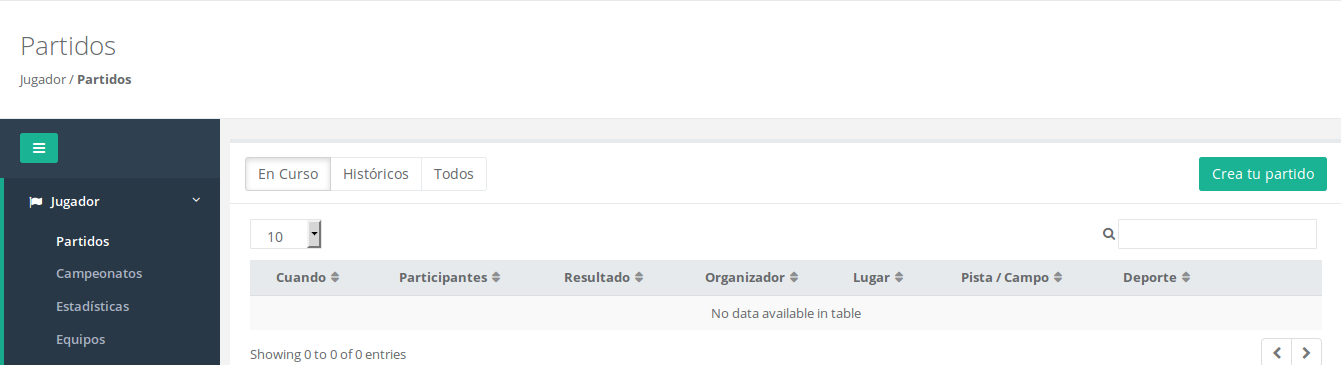
-Calendarizar partidos.

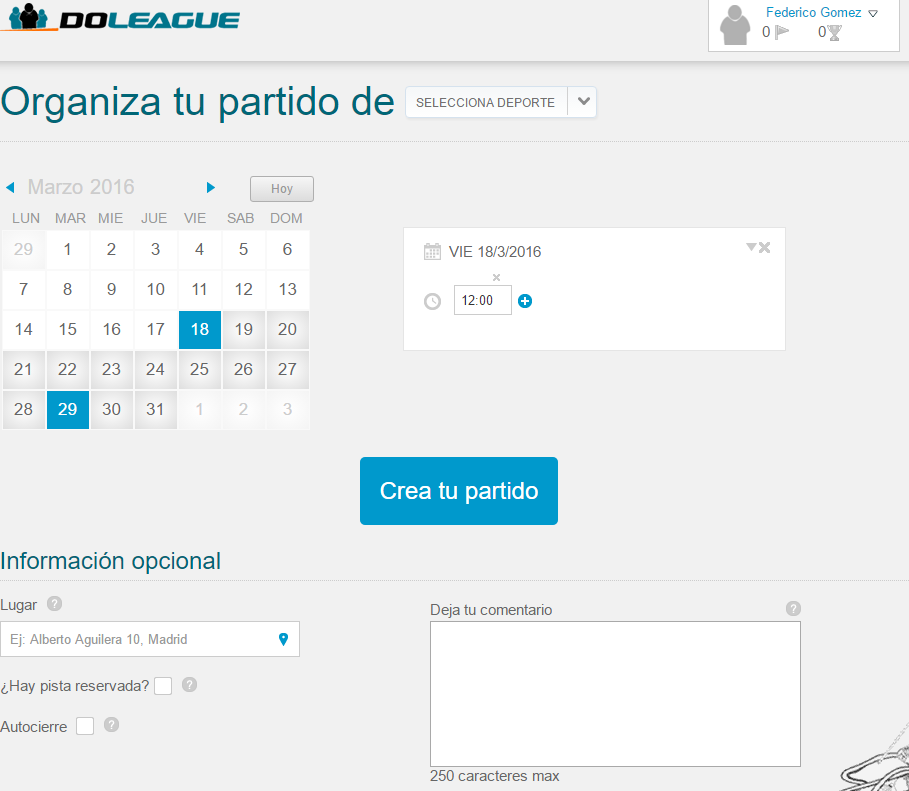
-Ofrece una aplicación mobile.

Funcionalidades destacadas

**Creación de partidos**

Permite crear un partido, elegir un horario y lugar pero este no está predefinido, es decir no tiene en cuenta ninguna cancha pre-cargada, solo escribir el nombre o ubicación, ni siquiera buscar la ubicación en Google Maps.





**Invitar amigos**

La única manera de invitar amigos es accediendo a través del link brindado una vez creado el partido. No permite invitar a tus amigos de Facebook, ni Twitter, ni Gmail. Solamente da la opción de compartir el link en las diferentes redes sociales.



**Historial de partidos y campeonatos**

En la pantalla principal se cuenta con un botón que permite mirar el histórico de los partidos, esta misma funcionalidad está disponible para campeonatos.

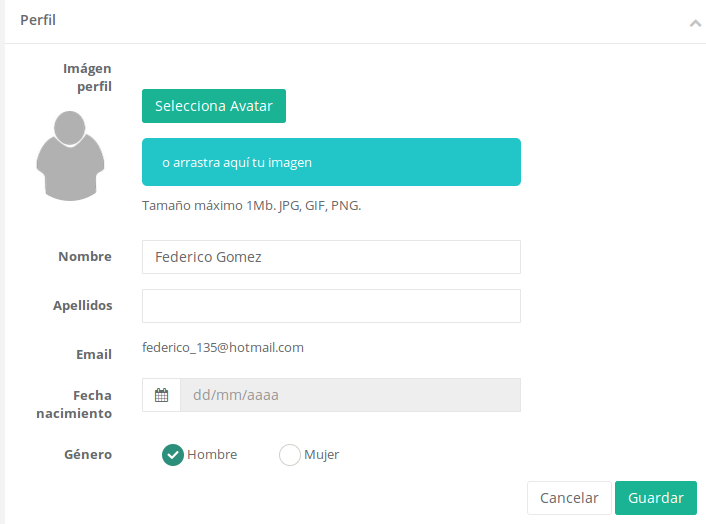


**Equipos y estadísticas**

Cuenta además con dos funciones para estadísticas y equipos pero estas opciones no están disponibles, según el sitio web serán añadidas próximamente.

**Perfil**

Lo único con lo que cuenta el perfil de nuestra cuenta es con lo que se muestra en la siguiente imagen.



Resumen de las funcionalidades detectadas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Características/Aplicación | Fulbo | Fubles | Calecio | DoLeague |
| Aplicación Web | **X** | **X** | **X** | **X** |
| Aplicación para Android |  | **X** | **X** | **X** |
| Aplicación para OS |  | **X** |  |  |
| Aplicación para Windows Phone |  | **X** |  |  |
| Registro con Facebook | **X** | **X** | **X** |  |
| Registro con Correo |  | **X** | **X** | **x** |
| Módulo “Reservar Cancha” |  |  |  |  |
| Módulo “Crear Partido” | **X** | **X** | **X** | **X** |
| Módulo “Partidos” | **X** | **X** | **X** | **X** |
| Módulo “Perfil Jugador” |  | **X** |  |  |
| Módulo “Pagos” |  |  |  | **X** |
| Módulo “Manejo de Notificaciones” |  |  | **X** |  |
| Notificaciones por Correo |  | **X** | **X** | **X** |
| Utilización GPS para saber la zona geográfica |  | **X** |  |  |

DESPEGAR

C:\Users\AdrielSuarez\Desktop\relevamiento\despegar\encabe.png

Link: <http://www.despegar.com.ar>

Descripción:

Despegar.com es una agencia de viajes *online*. Cuenta con presencia en 21 países. Despegar es el agencia de viajes número uno de América en volumen. Cuenta con reservas para hoteles, vuelos, traslados, cruceros, paquetes turísticos y más.

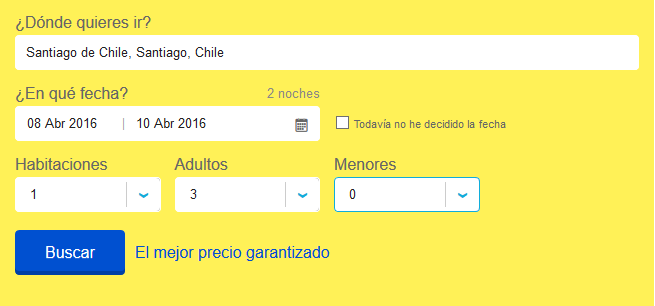
Características:

Permite encontrar las mejores ofertas en nuestras búsquedas en más de 180 países, con pago en destino o a través del sitio web. Ofrece una reserva fácil, online, pago en cuotas, cancelación sin cargo y rapidez en su búsqueda. También se puede administrar las reservas de forma online y cuenta con una aplicación para iOS y Android.

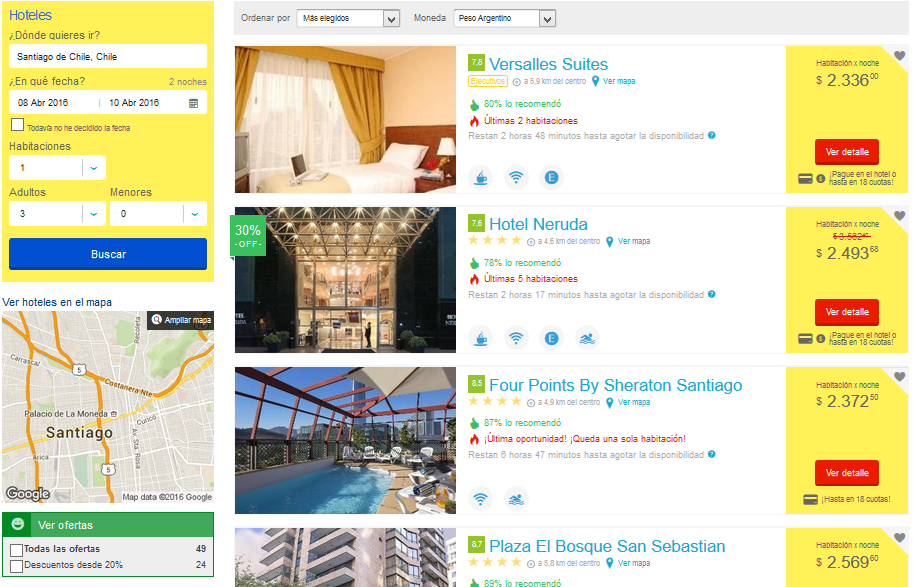
Funcionalidades destacadas:

**Reservas**

Ya sea que queramos realizar una reserva para hoteles, vuelos, cruceros, etc. Lo primero que debemos hacer es llenar una serie de datos solicitados así el sistema de despegar nos conduce a las mejores ofertas y disponibilidades para la fecha y destino seleccionados.

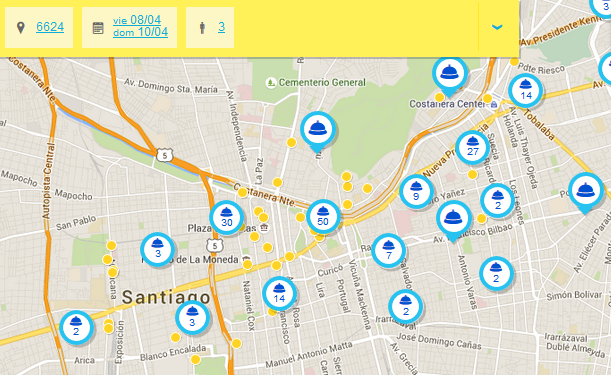
****

Nos permite visualizar en forma de listado las diferentes variantes que tenemos a la hora de realizar la reserva, cuenta con varios filtros para personalizar la búsqueda (mejores calificaciones, precios, alternativas de pago, servicios, entre otros).



**Zona geográfica**

Podemos visualizar nuestras búsquedas en el mapa por zona geográfica que nos ofrece a través de una api de Google Maps. Nos muestra la cantidad de puntos encontrados con las disponibilidades (en el caso de los hoteles) para la fecha seleccionada.



**Mi reserva**

Una funcionalidad interesante es la de “mi reserva”, donde nos permite visualizar todas las reservas activas que tenemos hasta el día de la fecha, nos permite consultar el estado de las mismas, solicitar modificaciones, cancelaciones, reenvio de vouchers y tickets así como también descargar tu factura.



Ingresando nuestro email de compra, nos envía por correo todas las reservas con las que contamos y la información de las mismas. También se puede logear a través de Facebook o Google.

C:\Users\AdrielSuarez\Desktop\relevamiento\despegar\a.png

En el caso de no haber reservas activas nos notifica mediante el mismo medio a nuestra cuenta.



MERCADO LIBRE



Link: www.mercadolibre.com

Descripción:

MercadoLibre, es una empresa argentina dedicada a la intermediación entre usuarios inscritos a su servicio de compras, ventas, pagos y subastas por Internet.

Los usuarios pueden vender tanto productos nuevos como usados a precio fijo o en la modalidad de [subastas](https://es.wikipedia.org/wiki/Subastas). MercadoLibre también posee [MercadoPago](http://www.mercadopago.com), una compañía que ofrece diversas modalidades de cobro a los vendedores. Sus oficinas centrales se encuentran en [Buenos Aires](https://es.wikipedia.org/wiki/Buenos_Aires), [Argentina](https://es.wikipedia.org/wiki/Argentina).

Características:

MercadoLibre tiene dos opciones para publicar:

* Básica (gratuita): dependiendo del tipo de publicación realizada puede o no tener comisión por venta
* Normal (con costo): que permite destacar productos, con comisión por venta.

Los compradores sólo deben registrarse en el sitio, buscar los productos o servicios que necesitan y hacer clic en el botón "comprar”. Vendedor y comprador reciben los datos de su contraparte vía [correo electrónico](https://es.wikipedia.org/wiki/Correo_electr%C3%B3nico) para que se contacten y perfeccionen la transacción.

Luego, ambos pueden calificarse para contarle al resto de la comunidad de usuarios cual fue su experiencia en cuanto a su contraparte, el producto y la transacción. El sistema de calificaciones permite a los compradores conocer la trayectoria de los vendedores dentro de sitio. El Sistema de Reputación de MercadoLibre cambió durante el año [2009](https://es.wikipedia.org/wiki/2009) para permitir una más fácil e intuitiva interpretación por parte de los usuarios. Los cambios fueron definidos en función de tests de usabilidad realizados con usuarios reales y con distintos niveles de experiencia dentro del sitio.

Funcionalidades Destacadas:

**Perfiles**

MercadoLibre cuenta con un sistema de reputación de usuarios fácil de usar y de consultar. Este Sistema es actualizado periódicamente conforme los usuarios efectúen transacciones en el sitio.

El sistema de reputación refleja la trayectoria de los usuarios, permitiendo diferenciar el comportamiento de cada uno como vendedor, como comprador y como usuario de los servicios que brinda MercadoLibre.

Para conocer la **reputación como comprador debes ingresar en la página de su perfil**, allí verás las calificaciones y comentarios recibidos de parte de los vendedores de la comunidad que han operado con él.



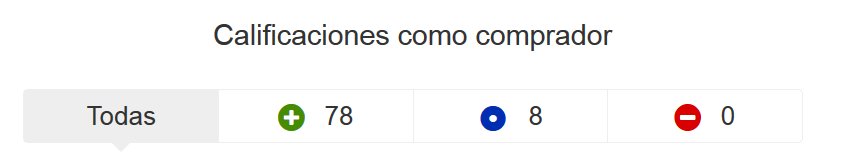
**Reputación como comprador**

La reputación como **comprador se determina en virtud de las calificaciones recibidas por otros usuarios** de la comunidad.

Cada vez que dos usuarios realizan una operación en el sitio tienen un plazo de 21 días para calificarse de forma positiva, negativa o neutral según haya sido su experiencia. Al mismo tiempo pueden dejar un comentario donde le cuentan al resto de la comunidad cómo fue operar con la otra parte.

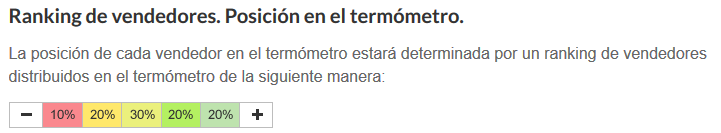
Calificación Positiva Suma 1 punto.  
Calificación Neutral No suma ni resta puntos.  
Calificación Negativa Resta 1 punto.

Un usuario podrá recibir tantas calificaciones como operaciones haya efectuado. Sin embargo, a los fines de establecer la reputación se tendrán en cuenta hasta tres calificaciones recibidas por la misma contraparte en un periodo de 90 días.

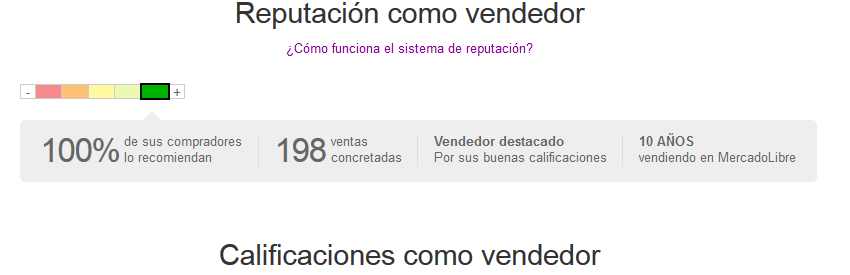


**Reputación como vendedor**

**La reputación de los vendedores se representará a través de un termómetro**, que se mostrará en todas las publicaciones.

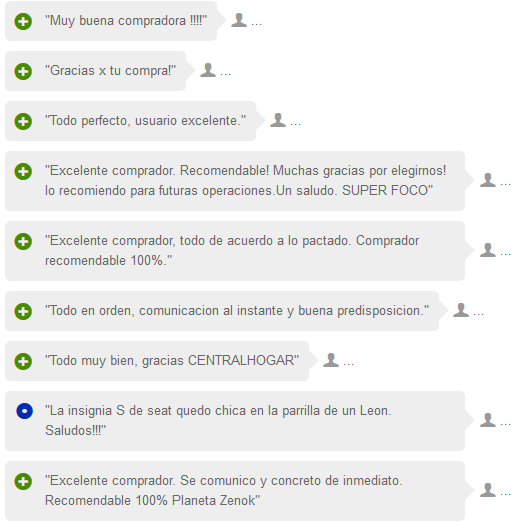


La reputación se determina automáticamente y consolida datos objetivos como: las operaciones llevadas a cabo en el sitio; la frecuencia con que el usuario vende en el sitio; las calificaciones recibidas de otros miembros de la comunidad; y el volumen en dinero de las ventas concretadas.



**Comentarios**

El sistema de calificaciones permite reflejar la trayectoria de los usuarios según lo informado por otros miembros de la comunidad, incluyendo comentarios y réplicas realizados éstos bajo su exclusiva responsabilidad. MercadoLibre no controla los comentarios ni verifica la veracidad o exactitud de los mismos y no se responsabiliza por lo allí expresado, por las transacciones que los usuarios realicen o no teniéndolos en cuenta, por la confianza depositada en las calificaciones de la contraparte o por otros comentarios expresados dentro del sitio o a través de cualquier otro medio, incluido el correo electrónico.



**Restricciones del sistema**

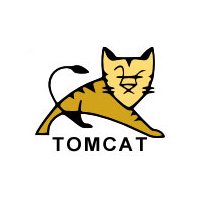
Está prohibido hacer comentarios y expresar opiniones de carácter privado o personal, que no tengan relación con la operación sobre la cual se está calificando a otro usuario. MercadoLibre podrá restringir y/o prohibir la contratación de cualquiera de sus servicios a los usuarios cuya reputación esté por debajo de ciertos niveles fijados por MercadoLibre a su exclusiva discreción

Tecnología de Información

A continuación se detallan algunas tecnologías utilizadas por las aplicaciones antes relevadas, además algunas de ellas con herramientas y servicios que pueden utilizarse a la hora de desarrollar una aplicación web.

Se explicará Tomcat, JSP (JavaServer Pages), arquitectura Cliente-Servidor y MVC (modelo-vista-controlador), CSS3, Less.js, Photoshop, HTML5, JavaScript, Knockout js, Java, JQuery, Boostrap Symfony, PHP, Android Java SDK, Facebook, Whatsapp.

Tomcat



Apache Tomcat (también llamado Jakarta Tomcat o simplemente Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation (aunque creado por Sun Microsystems).

Tomcat es desarrollado y actualizado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 8.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Tomcat es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

JSP (JavaServer Pages)



Es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

Para desplegar y correr JavaServer Pages, se requiere un servidor web compatible con contenedores [servlet](https://es.wikipedia.org/wiki/Servlet) como [Apache Tomcat](https://es.wikipedia.org/wiki/Apache_Tomcat) o [Jetty](https://es.wikipedia.org/wiki/Jetty).

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el [PHP](https://es.wikipedia.org/wiki/PHP), [ASP](https://es.wikipedia.org/wiki/Active_Server_Pages) o los [CGIs](https://es.wikipedia.org/wiki/Interfaz_de_entrada_com%C3%BAn), programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas.

Para empezar, los JSPs y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él. Cada servlet (o JSP) se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa + [intérprete](https://es.wikipedia.org/wiki/Int%C3%A9rprete_%28inform%C3%A1tica%29)). Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados [servidores](https://es.wikipedia.org/wiki/Servidor), y los demandantes, llamados [clientes](https://es.wikipedia.org/wiki/Cliente_%28inform%C3%A1tica%29). Un cliente realiza peticiones a otro programa, el [servidor](https://es.wikipedia.org/wiki/Servidor), quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo [multiusuario](https://es.wikipedia.org/wiki/Multiusuario) distribuido a través de una [red de computadoras](https://es.wikipedia.org/wiki/Red_de_computadoras).

Algunos ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son el [Correo electrónico](https://es.wikipedia.org/wiki/Correo_electr%C3%B3nico), un [Servidor de impresión](https://es.wikipedia.org/wiki/Servidor_de_impresi%C3%B3n) y la [World Wide Web](https://es.wikipedia.org/wiki/World_Wide_Web).

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre [cliente](https://es.wikipedia.org/wiki/Cliente_%28inform%C3%A1tica%29) y [servidor](https://es.wikipedia.org/wiki/Servidor) es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de [servidores](https://es.wikipedia.org/wiki/Servidor) incluyen los servidores [web](https://es.wikipedia.org/wiki/World_Wide_Web), los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los *sistemas multicapa* en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes [computadoras](https://es.wikipedia.org/wiki/Computadora_electr%C3%B3nica) aumentando así el grado de distribución del sistema.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

Características

En la arquitectura C/S el remitente de una solicitud es conocido como [cliente](https://es.wikipedia.org/wiki/Cliente_%28inform%C3%A1tica%29). Sus características son:

* Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
* Espera y recibe las respuestas del servidor.
* Por lo general, puede conectarse a varios servidores a la vez.
* Normalmente interactúa directamente con los usuarios finales mediante una [interfaz gráfica de usuario](https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario).

Al receptor de la solicitud enviada por el cliente se conoce como [servidor](https://es.wikipedia.org/wiki/Servidor). Sus características son:

* Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
* Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
* Por lo general, acepta las conexiones de un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).

En la arquitectura C/S sus características generales son:

* El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
* Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
* Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
* La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
* Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización.

Los servidores pueden ser apátridas o stateful. Un servidor apátrida no guarda ninguna información entre las peticiones. Un servidor stateful puede recordar la información entre las peticiones. El alcance de esta información puede ser global o sesión-específico. Un servidor del [HTTP](https://es.wikipedia.org/wiki/HTTP) para las páginas estáticas del HTML es un ejemplo de un servidor apátrida mientras que [Apache Tomcat](https://es.wikipedia.org/wiki/Apache_Tomcat) es un ejemplo de un [servidor](https://es.wikipedia.org/wiki/Servidor) stateful.

Ventajas

* Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes [P2P](https://es.wikipedia.org/wiki/P2P)).
* [Escalabilidad](https://es.wikipedia.org/wiki/Escalabilidad): se puede aumentar la capacidad de [clientes](https://es.wikipedia.org/wiki/Cliente_%28inform%C3%A1tica%29) y [servidores](https://es.wikipedia.org/wiki/Servidor) por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
* Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como [encapsulación](https://es.wikipedia.org/wiki/Encapsulaci%C3%B3n).
* Existen [tecnologías](https://es.wikipedia.org/wiki/Tecnolog%C3%ADa), suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la [seguridad](https://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica) en las [transacciones](https://es.wikipedia.org/w/index.php?title=Transacci%C3%B3n_%28red%29&action=edit&redlink=1), la amigabilidad de la [interfaz](https://es.wikipedia.org/wiki/Interfaz), y la facilidad de empleo.

Desventajas

* La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultáneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes [P2P](https://es.wikipedia.org/wiki/P2P) como cada [nodo](https://es.wikipedia.org/wiki/Nodo_%28inform%C3%A1tica%29) en la red hace también de servidor, cuanto más nodos hay, mejor es el [ancho de banda](https://es.wikipedia.org/wiki/Ancho_de_banda) que se tiene.
* El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está *caído*, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.
* El [software](https://es.wikipedia.org/wiki/Software) y el [hardware](https://es.wikipedia.org/wiki/Hardware) de un servidor son generalmente muy determinantes. Un hardware regular de un [ordenador personal](https://es.wikipedia.org/wiki/Ordenador_personal) puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.
* El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la [aplicación es una Web](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web), no podemos escribir en el disco duro del cliente o imprimir directamente sobre las [impresoras](https://es.wikipedia.org/wiki/Impresora) sin sacar antes la ventana previa de impresión de los navegadores.

Arquitectura MVC (Modela-Vista-Controlador).



El modelo-vista-controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

Modelo

Contiene el núcleo de la funcionalidad (dominio) de la aplicación.

Encapsula el estado de la aplicación.

No sabe nada / independiente del Controlador y la Vista.

Vista

Es la presentación del Modelo.

Puede acceder al Modelo pero nunca cambiar su estado.

Puede ser notificada cuando hay un cambio de estado en el Modelo.

Controlador

Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente.

CSS3



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 XML2 (y por extensión en XHTML). El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML.En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo «style».

Less



Es un lenguaje de hojas de estilo dinámico que puede ser compilado con CSS y ejecutado del lado del cliente o del servidor. Es de código abierto. Su primera versión fue escrita en Ruby, pero actualmente sus nuevas versiones están hechas en JavaScript. La sintaxis de Less pertenece a un metalenguaje aninado, en este caso el código válido para CSS es válido para Less ya que poseen la misma semántica. La principal diferencia entre Less y otros pre compiladores de CSS es que Less permite la pre compilación en tiempo real vía less.js en el navegador.

Photoshop



Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente "taller de fotos".

Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general.

Actualmente forma parte de la familia Adobe Creative Suite y es desarrollado y comercializado por Adobe Systems Incorporated inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows. Su distribución viene en diferentes resentaciones, que van desde su forma individual hasta como parte de un paquete, siendo estos: Adobe Creative Suite Design Premium y Versión Standard, Adobe Creative Suite Web Premium, Adobe Creative Suite Production Studio Premium y Adobe Creative Suite Master Collection.

HTML



HTML, siglas de HyperText Markup Language («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

**TRABAJO PRÁCTICO N°2: Planificación de Proyectos Informáticos**

CAPITULO I**: Actividades**

**Definición y descripción de actividades.**

* Selección de Organización/Empresa o Sistema (Fecha límite → 15/03)

Se debe elegir un sistema para desarrollar en la materia Proyecto Final (UTN FRM 2016).

El mismo debe ser uno donde se pueda implementar los conocimientos adquiridos durante el cursado y, a su vez, poder ser implementado en la vida real.

Para ello:

* Formulamos propuestas.
* Seleccionamos las propuestas más interesantes.
* Presentamos las propuestas ante la cátedra.
* Definición de Requerimientos (Fecha límite → 03/05)

Para poder conocer el entorno y saber lo que realmente se necesita, investigamos sobre sistemas que tienen funcionalidades similares o bien, algún módulo que sea de nuestro interés porque refleja un comportamiento similar a alguno nuestro y puede llegar a servirnos, a la hora de implementar, nuestro sistema.

Cabe destacar que muchos sistemas a analizar no tienen el mismo objetivo que el sistema propuesto.

Para ello, realizamos:

* Relevamiento general → (22/03-29/03)
* Investigar empresas, organizaciones y páginas con temáticas similares.
* Realizar entrevistas a personas involucradas.
* Documentar entrevistas.
* Relevar detalles e historia de los sistemas encontrados.
* Relevar funciones detectadas de los sistemas.
* Relevar tecnologías de la información de los sistemas.

CAPITULO II: **Organización para la ejecución del proyecto.**

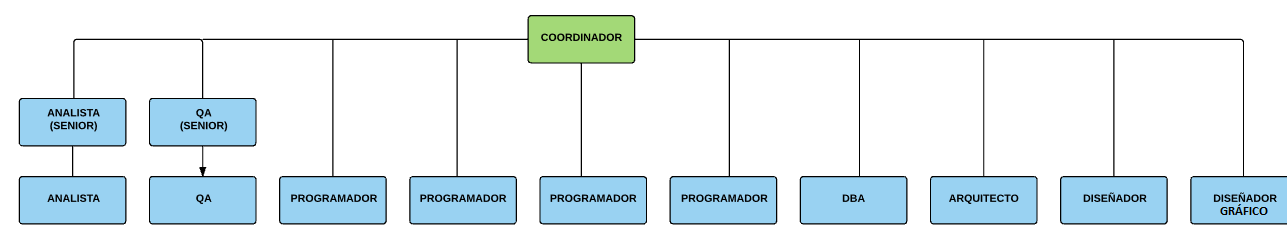
Equipo de trabajo

**Estructura**

A continuación definiremos la estructura de nuestro equipo de trabajo que consideramos adecuada para el desarrollo del proyecto. Contamos con un coordinador del equipo el cual está encargado de controlar que las tareas llevadas a cabo por los miembros del grupo sean acordes y se cumplan en tiempo y forma.

Todos los miembros del equipo son responsables de establecer los objetivos, tomar decisiones, aportar sus conocimientos y habilidades para la solución de problemas, fijando las tareas necesarias y midiendo el desempeño personal.

En este organigrama podemos contemplar la estructura del equipo de trabajo.



**Puestos, perfiles y cantidades**

Conocimiento y experiencia requerida: faltan resposabilidades

* Coordinador (1).
* Profesional con título en Ingeniería en Sistemas de Información o afín.
* Avanzado nivel en idioma inglés.
* Experiencia mínima en el área de 3 años.
* Habilidad de ejercer su autoridad como líder.
* Sólidos conocimiento y experiencia en las herramientas vinculadas a la gestión  
  de proyectos.
* Experiencia en armado y coordinación de equipos de trabajo.
* Experiencia y predisposición para capacitación de personal.
* Capacidad de comunicación.
* Analista (2).
* Profesional con título en Ingeniería en Sistemas de Información o relacionado.
* Avanzado nivel en idioma inglés.
* Experiencia mínima en el área de: 1 año (junior), 3 años (semi-senior), 5 años (senior).
* Conocimiento de las metodologías de desarrollo de software.
* Entendimiento y planificación del negocio y su impacto organizacional, ambiental y humano.
* Conocimiento de las herramientas necesarias para la gestión de calidad.
* Conocimientos Mediciones y Métricas.
* Capacidad de comunicación.
* QA (2).
* Profesional con título en sistemas de información y/o relacionados o estudiante avanzado.
* Experiencia mínima en el área de: 1 año (junior), 3 años (semi-senior), 5 años (senior).
* Avanzado nivel en idioma inglés.
* Conocimiento de herramientas para la ejecución y seguimiento de testing.
* Capacidad para el diseño y análisis de casos de prueba.
* Facilidad para la comunicación.
* Entendimiento del negocio y criterios de aceptación.
* Programador (4).
* Profesional con título en sistemas de información y/o afín o estudiante avanzado.
* Experiencia mínima en el área de: 1 año (junior), 3 años (semi-senior), 5 años (senior).
* Conocimiento intermedio o avanzado del idioma inglés.
* Capacidad de aprendizaje.
* Ganas de capacitarse
* Capacidad para trabajo en equipo.
* Conocimientos teórico-prácticos del desarrollo y administración de servicios web.
* Proactividad y pensamiento crítico.
* Sólidos conocimientos teóricos en JavaScript.
* Administrador de base de datos “DBA” (1).
* Profesional con título en Ingeniería en Sistemas de Información o relacionado a Administración de Base de Datos.
* Avanzado nivel en idioma inglés.
* Sólidos conocimientos teóricos – prácticos del modelo y algebra relacional
* Sólidos conocimientos teóricos en Mongo DB.
* Control de Acceso y administración de privilegios.
* Conocimientos de DataWharehousing y DataMining.
* Experiencia en sistemas de respaldo de bases de datos.
* Arquitecto (1).
* Profesional con título en sistemas de información y/o relacionados.
* Conocimiento intermedio o avanzado del idioma inglés.
* Capacidad para experimentar con distintos enfoques arquitectónicos.
* Conocimientos para documentar modelos y componentes y especificaciones de interfaces.
* Sólidos conocimientos de programación, análisis y diseño: estructurado y orientado a objetos
* Sólidos conocimientos y experiencia laboral con patrones de software empresarial,  
  patrones de integración y patrones de arquitectura empresarial.
* Diseñador (1).
* Profesional con título en Ingeniería en Sistemas de Información o relacionado.
* Avanzado nivel en idioma inglés.
* Conocimientos en tecnologías a utilizar.
* Sólidos conocimientos teóricos en la aplicación de patrones de diseño.
* Conocimientos teóricos para la medición de la performance de las aplicaciones.
* Pensamiento de Integración.
* Capacidad de análisis y de síntesis para la resolución de problemas complejos.
* Facilidad para la comunicación.
* Diseñador Gráfico (1).
* Profesional con título en Diseño Gráfico o relacionado.
* Habilidad para comunicar e interpretar conceptos visualmente.
* Capacidad de análisis y síntesis.
* Conocimientos en comunicación visual.
* Conocimiento en tecnologías web.
* Facilidad para el dibujo e ilustración.
* Capacidad para generar maquetas y prototipos rápidos.

**Funciones principales de los miembros del equipo de trabajo.**

|  |  |
| --- | --- |
| MIEMBRO | ROLES |
| Álvarez, Martín | Analista , QA , Programador |
| García, Mauricio | Analista, DBA, Programador |
| Gómez, Federico | Arquitecto, QA , Programador |
| Suarez, Adriel | Coordinador, Diseñador, Programador |

Modificar tabla x tablas cruces.

Métodos de comunicación formal, control de avance, retroalimentación, decisiones

Todos los miembros serán responsables de conocer la planificación establecida por el grupo y las próximas tareas a realizar, para cumplir con los tiempos establecidos. En base a esto, utilizaremos una herramienta de integración continua denominada ……….... Todos los miembros tendrán permisos de crear tareas y solo el coordinador será el encargado de asignarlas a otros miembros o así mismo.

Semanalmente, los sábados por la mañana, nos comunicaremos vía Skype, personalmente o mediante Whatsapp, dependiendo de la disponibilidad de los miembros, para informar lo realizado en relación a la tarea asignada, dificultades encontradas o todo tipo de información relacionada al desarrollo del proyecto. En función a dicha reunión, se tomarán decisiones sobre los pasos a seguir para cumplir con la planificación establecida.

Los días martes por la mañana, se realizará otra reunión, con las mismas condiciones, para charlar sobre los avances realizados en función a lo establecido en la reunión anterior y preparar el material correspondiente para recibir la retroalimentación por parte del profesor correspondiente a la cátedra. Una vez finalizada la retroalimentación, se anotarán las observaciones en su lugar correspondiente y luego, se subirán a la nube utilizada para la disponibilidad de todos los miembros. Los días miércoles por la mañana, se establecerá un nuevo contacto para determinar los pasos a seguir en función de la retroalimentación obtenida.

Los cambios que se realicen tanto en el código como en otros entregables y elementos intermedios, serán subidos al repositorio correspondiente. Cada miembro será responsable de actualizarlo periódicamente, para que los demás miembros del equipo estén al tanto del trabajo.

Gestión de configuración del Software

Desde nuestra posición se considera que la gestión de configuración es vital para asegurar el éxito y la calidad del proyecto. Se debe llevar a cabo un estricto control de los cambios que se van produciendo a lo largo del desarrollo del Sistema de Información, tanto del código en sí como de la documentación, planificación y todo lo relacionado al sistema en sí. Se debe asegurar también que siempre haya una versión estable y disponible para todas las personas involucradas en el desarrollo del sistema. Al estar todos involucrados, todos serán responsables de realizarla correctamente.

Los elementos de configuración (es decir, aquellos productos intermedios o finales cuyos cambios se van a gestionar) que tendremos en cuenta son:

**Elementos que evolucionan en el tiempo:**

* Código fuente.
* Assets (Imágenes, fuentes y elementos relacionados).
* Requerimientos.
* Modelos (de datos, de diseño, entre otros).
* Manuales de usuario.
* Planes (de proyecto, de pruebas, entre otros).
* Casos de prueba, y resultados de pruebas.
* Informes.

**Elementos invariables en el tiempo:**(que, aunque no cambien, deben estar disponibles para todos)

* Minutas sobre devoluciones recibidas en clases.
* Minutas de reuniones.
* Documentación proporcionada por la cátedra de Proyecto Final.
* Paquetes de entregables/prototipos.

**Herramientas a utilizar**

La gestión de la configuración se llevará a cabo mediante el uso de Git como sistema de almacenamiento, que además es un sistema de control de versiones descentralizado. El repositorio estará subido a GitHub, de manera de estar siempre disponible para todos.

“GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.”

Fuente: <https://es.wikipedia.org/wiki/GitHub>.

**Estructura/Método**

Se alojará en GitHub un repositorio del cual todos los miembros podrán realizar *pull* y sobre el cual harán *push*. Cada miembro podrá clonar el repositorio y trabajar sobre él. Existirá un branch develop en el que el código que está siendo desarrollado será versionado. Cuando una porción de funcionalidad ha sido completada y está lista para ser probada según lo juzgue la persona que lo está desarrollando, se pasará al branch de testing. Aquellas funcionalidades probadas y operativas formarán parte del branch de master, cuyas versiones reflejan incrementos del sistema que pudieran ser entregables. Aquellas porciones cuyo testing arroje errores o inconsistencias, serán notificadas a los responsables para que las modifiquen en el branch develop para posteriormente volver a incluirlas en testing. Cada cambio que se realice sobre los branches irá acompañado de un comentario que refleje lo que se incluyó en ese cambio. Para los elementos de configuración se realizará un procedimiento similar que para el código, con la salvedad de que los elementos invariables en el tiempo pertenecerán directamente al branch master.

Todos los miembros actuarán como gestores de configuración, ya que todos tendrán los mismos permisos y responsabilidades.

Debido a la necesidad de adaptación a esta metodología de trabajo, tanto la gestión de configuración como los métodos formales de comunicación regirán a partir del 29/03, y se utilizará Google Docs como herramienta temporal.