Desarrollo de Software – Curso 2020-2021 - Examen primer parcial **9 abril 2021**

Nombre y apellidos:	Raúl Castro Moreno		
Núm. de grupo pequeño:	4	Núm. dentro del grupo pequeño (empezando por el 1)[1]:	1

Instrucciones para realizar el examen:

- Se debe utilizar este fichero para hacer el examen, contestando debajo de cada pregunta.
- Hay cuatro modalidades de este examen (A, B, C y D), correspondiendo cada modalidad a los estudiantes con el número de orden correlativo (1, 2, 3 y 4) dentro de su grupo pequeño.
- Se subirá a PRADO el examen en formato pdf, hasta el 9 de abril incluido.
- La tabla siguiente especifica las variantes de cada una de las preguntas según la modalidad de examen.

MODALIDADES DE EXAMEN

		# orden dentro del grupo (modalidad de examen)			
		1 (A)	2 (B)	3 (C)	4 (D)
# pregunta	1 (patrones de diseño)	*Builder*	*Fachada*	*Decorador*	*Estrategia*
	2 (estilos arquitectónicos)	*Tubería lineal*	*Manejador de eventos*[2]	*Basado en capas*[3]	*Pizarra*
	3 (descripción arquitectónica)	*Vista funcional*	*Perspectiva de seguridad*	*Vista contextual*	*Perspectiva de evolución*
	4 (pruebas software)	*Requisitos funcionales*	*Requisitos no funcionales*	*Requisitos funcionales*	*Requisitos no funcionales*

1. **Patrones de diseño (2,5 puntos)**: Modifica el supuesto práctico de tu grupo de teoría de forma que se pueda aplicar muy bien el patrón de diseño que te corresponde según la tabla. Debes incluir:
   - **(1 punto)** Una descripción en lenguaje natural del caso de estudio modificado en el que se vea la gran conveniencia de aplicar ese patrón.

Para el supuesto práctico de mi grupo de teoría, el cuál trata de una aplicación de encuentros deportivos, la mejor forma de utilizar el patrón de diseño Builder adaptado es utilizandolo para la creación de las Pistas de deporte las cuáles guarda la aplicación.

Ahora ya si, basándonos en como representaría este patrón de diseño sobre este caso de estudio, tendríamos las siguientes clases:

1. El número de orden, es el lugar que ocupa una persona en la lista de miembros del grupo pequeño, considerando el orden ascendente por apellidos (según aparecen los apellidos en PRADO).
2. También llamado *Publicar/suscribir* (invocación implícita).
3. Deben utilizarse al menos tres capas.

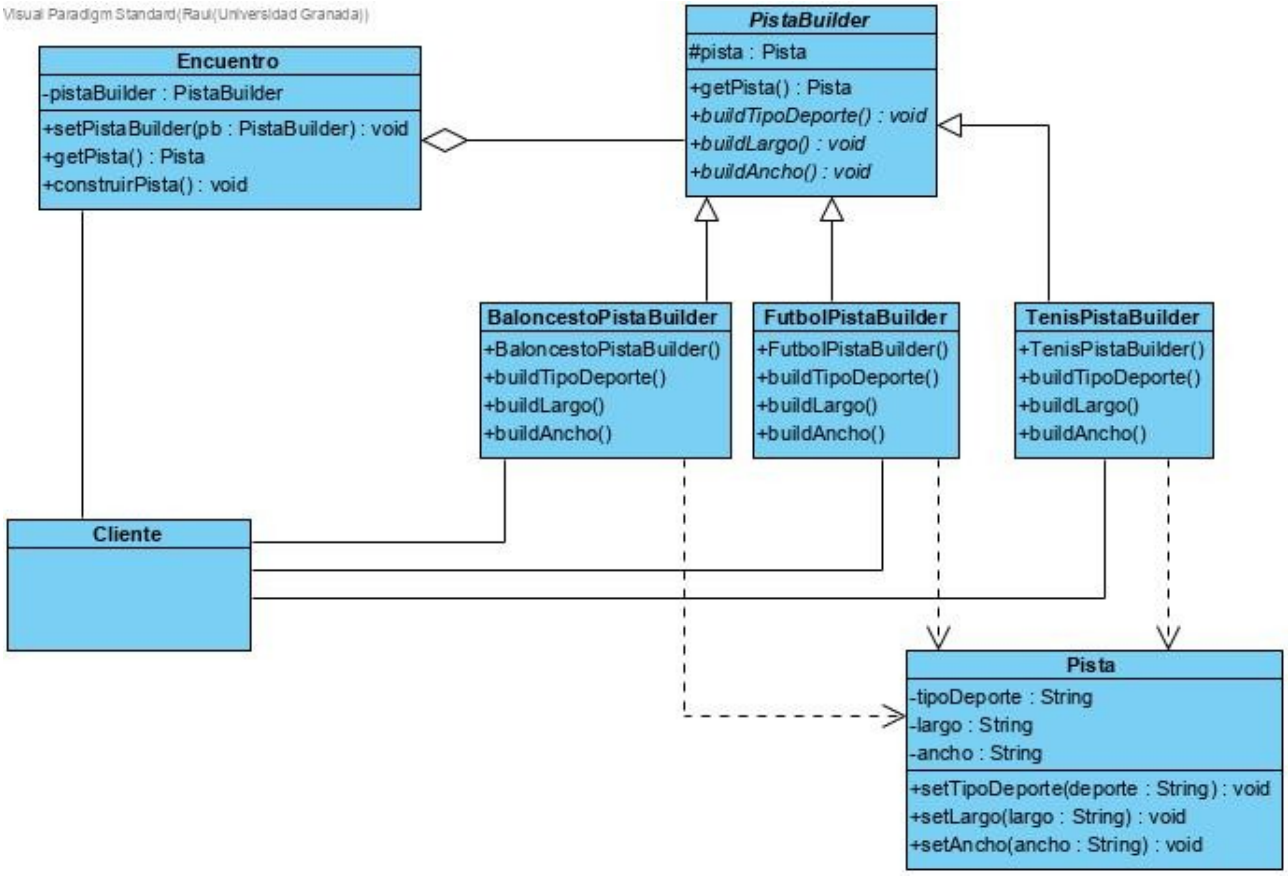
Como clase **Producto** del patrón, tendríamos la clase **Pista,** con los atributos privados de **tipo de deporte, largo y ancho.**

Como clase abstracta del Builder , sería la clase **PistaBuilder**, con un atributo protegido de una Pista, y los métodos abstractos publicos de **buildTipoDeporte()** , **buildLargo()** y **buildAncho()**, y también un método publico **getPista()** que nos devuelva la pista.

Como clases Builder concretas que extienden PistaBuilder ,tenemos **FutbolPistaBuilder** , **BaloncestoPistaBuilder** y **TenisPistaBuilder**, y en cada una utilizan un constructor y los métodos de **buildTipoDeporte()** , **buildLargo()** y **buildAncho()** con los valores específicos de cada deporte para un **Pista** . (Ejemplo: Tenis pondría el tipo de deporte como “Tenis”, y el largo y ancho, como las medidas de una pista de Tenis.)

Por último, como clase Director, sería la clase **Encuentro** , con un atributo privado de PistaBuilder, y un método de **setPistaBuilder()** donde establecemos el Builder concreto que queremos, otro método **getPista()** que devuelve la pista del **PistaBuilder** de la clase. Y un método **contruirPista()** que utiliza los 3 métodos build del atributo **PistaBuilder.**

- **(1,5 puntos)** El diagrama de clases de diseño donde se vea la aplicación del mismo, incluyendo una clase *Cliente* que usará los objetos gestionados por el patrón.



2. **Estilos arquitectónicos (2,5 puntos)**: Modifica el supuesto práctico de tu grupo de teoría de forma que se pueda aplicar muy bien el estilo arquitectónico que te corresponde según la tabla. Debes incluir:
   - **(1 punto)** Una descripción en lenguaje natural del caso de estudio modificado en el que se vea la gran conveniencia de aplicar ese estilo arquitectónico.

Para nuestro supuesto práctico de encuentros deportivos, la modificación que he pensado para la que convendría usar el estilo arquitectónico de Tubería Lineal, es tener los datos del Alquiler de una ubicación, y estos datos, pasarlos por unos filtros los cuales modifiquen el coste de los alquileres dependiendo de las personas que vayan a usarlo, el tiempo que vayan a utilizarla, y por último aplicarle el IVA.

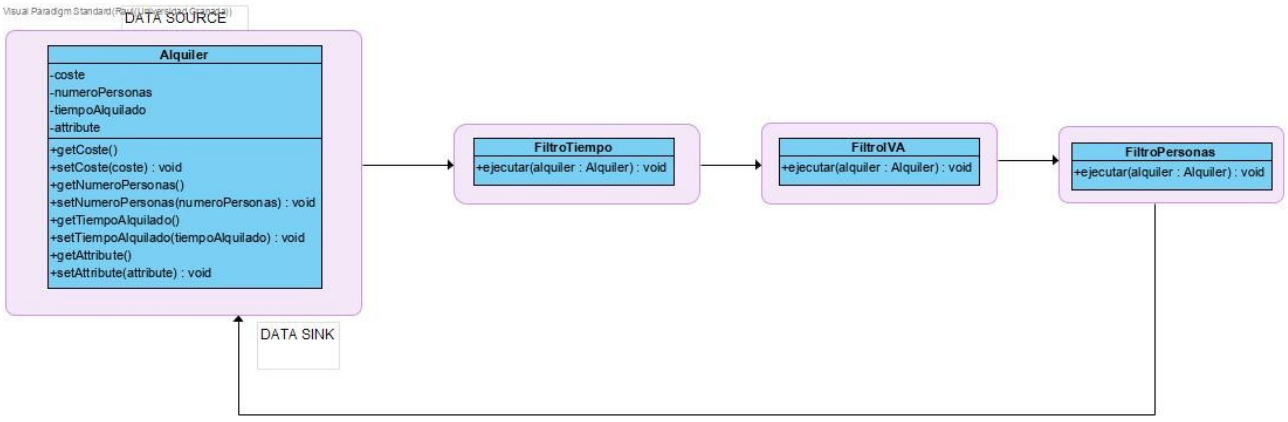
Por tanto, tendríamos la clase **Alquiler**, con sus datos privados de **coste**, **tiempoAlquilado**, **numeroPersonas** . Tendría tambíen sus métodos de get y set de los atributos.

Luego tendríamos la clase **FiltroTiempo,** que dependiendo del tiempo de alquiler, modifica el coste, donde a cuanto más tiempo, más aumenta el coste.

Otra clase filtro llamada **FiltroPersonas,** que con los datos filtrados por la clase FiltroTiempo, a cuantas más personas haya en el alquiler, se aplica un pequeño descuento.

Por último, después de esos filtros, esos datos pasan por la clase **FiltroIVA,** que le aplica el impuesto de IVA que haya vigente, en este caso el 21%.

   - **(1,5 puntos)** Un diagrama genérico, de bloques (*cuadros y líneas*) o un diagrama UML (de componentes, de interacción, de clases, de estado ...) donde se vea la aplicación del mismo.



3. **Descripción arquitectónica (DA) (4 puntos)**: A partir de la descripción de la pregunta 2, realiza los siguientes pasos para diseñar la arquitectura software de esa aplicación:
   - **(0,5 puntos)** Especifica los requisitos funcionales y no funcionales finales (seguramente los irás cambiando conforme avances en la DA)

Objetivos principales:

- Proveer de una aplicación para jugar a deportes con personas que no conoces por mero entretenimiento.

- Que usuarios una vez la prueben, se lleven una buena experiencia y quieran volver a usarla

Desarrollo de Software – Curso 2020-2021 - Examen primer parcial 9 abril 2021

Requisitos no Funcionales:

- **Optimizado:** La aplicación debe de dar respuestas muy rápidas en todas las secciones, ya que la lentitud al dar respuestas estropea la experiencia de usuario.

- **Seguro y Regulable:** Mediante mecánismos de protección, proteger los datos del sistema ya sea de la BD o datos de pagos, de ataques de hackers que vulneren la privacidad de los usuarios, cumpliendo así las leyes de protección de datos.

- **Fiable y Alta Disponibilidad:** La aplicación debe ser fiable en el sentido de caídas de servidores, presentando así una alta disponibilidad, para ello tener servidores auxiliares, que funcionen en caso de algún problema. También un control alto sobre posibles errores para intentar arreglarlos antes de que sucedan.

- **Escalable y Elástico :** Tenga la posibilidad de aumentar o disminuir las prestaciones del sistema en función de la carga que haya, y que se pueda hacer de forma automática (Elasticidad)

- **Intuitivo y fácil de entender :** Crear una interfaz que pueda usar cualquier persona, que se puede cambiar el tamaño de fuente, y varios tutoriales que expliquen rápido y fácil el funcionamiento.

Requisitos Funcionales:

- **Alta Usuario**
- **Baja Usuario**
- **Consultar Datos Usuario**
- **Alta Ubicacion**
- **Baja Ubicacion**
- **Consultar Datos Ubicacion**
- **Crear Encuentro**
- **Mostrar Lista Encuentros**
- **Inscribir Usuario a Encuentro**

Detalle	Descripción
RF#	1
Nombre	Alta Usuario
Descripción	Permite el alta de un usuario en la aplicación
Entrada	Datos del usuario (nickname , email , contraseña, nombre, edad, teléfono)
Procesamiento	[Prerrequisito: acceso no identificado] Registro de la cuenta en la BD externa al sistema ; [Postrrequisito: acceso identificado]
Salida	Confirmación del alta

Detalle	Descripción
RF#	2
Nombre	Baja Usuario
Descripción	Permite darse de baja a un usuario de la aplicación
Entrada	Nickname y contraseña
Procesamiento	[Prerrequisito: acceso identificado] Borrado de la cuenta en la BD externa al sistema
Salida	Confirmación de la baja

Detalle	Descripción
RF#	3
Nombre	Consultar Datos Usuario
Descripción	Permite consultar los datos de un usuario registrado en la aplicación
Entrada	El nickname de un usuario
Procesamiento	[Prerrequisito: acceso identificado] Consulta del usuario identificado por el nickname en la BD externa al sistema
Salida	Detalles del usuario (nombre, edad, email, teléfono)

Detalle	Descripción
RF#	4
Nombre	Alta Ubicación
Descripción	Permite añadir ubicaciones para usar en los encuentros al sistema
Entrada	Datos de la ubicación (nombre ,localización, medidas, tipo de suelo, deporte, fotos)
Procesamiento	[Prerrequisito: acceso identificado como gestor de la aplicación] Registro de la ubicación en la BD externa al sistema
Salida	Confirmación de la alta.

Detalle	Descripción
RF#	5
Nombre	Baja Ubicación
Descripción	Permite borrar la ubicación del sistema
Entrada	Nombre de la ubicación
Procesamiento	[Prerrequisito: acceso identificado como gestor de la aplicación] Borrado de la ubicación en la BD externa al sistema
Salida	Confirmación de la baja

Detalle	Descripción
RF#	6
Nombre	Consultar Datos Ubicación
Descripción	Permite consultar los datos de una ubicación registrado en la aplicación
Entrada	El nombre de una ubicación
Procesamiento	[Prerrequisito: acceso identificado] Consulta la ubicación identificado por el nombre en la BD externa al sistema
Salida	Detalles de la ubicacion (localización, medidas, tipo de suelo, deporte)

Detalle	Descripción
RF#	7
Nombre	Crear Encuentro
Descripción	Permite a un usuario crear un encuentro deportivo
Entrada	Nickname usuario, nombre de la ubicación y hora
Procesamiento	[Prerrequisito: acceso identificado, la ubicación debe estar disponible] Se crea un encuentro en la BD externa con los datos del usuario que lo crea.
Salida	Confirmación de la creación de encuentro y código identificativo del encuentro

Detalle	Descripción
RF#	8
Nombre	Mostrar Lista Encuentros
Descripción	Muestra una lista de los encuentros disponibles
Entrada	
Procesamiento	[Prerrequisito: acceso identificado] Consulta y lista los datos de los encuentros de la BD externa.
Salida	Lista con todos los encuentros disponibles

Detalle	Descripción
RF#	9
Nombre	Inscribir Usuario a Encuentro
Descripción	Permite a un usuario inscribirse a un encuentro
Entrada	Código del encuentro
Procesamiento	[Prerrequisito: acceso identificado] Se añaden los datos del usuario al encuentro identificado por el código en la BD externa
Salida	Confirmación de la inscripción

- **(0,5 puntos)** Especifica las partes interesadas en el sistema y sus intereses

- Listado y descripcion de las partes interesadas en el sistema

Arquitecto: (master en ingeniera informatica) es la persona que se le ocurrió la idea de la aplicación y creador de la empresa que lleva a cabo la creación de esta aplicación. Sera el arquitecto software encargado de elaborar la D.A. y supervisar que se desarrolle de forma apropiada.

Desarrollador y técnico de pruebas: es un desarrollador (grado superior de FP en desarrollo de aplicaciones para móvil) que trabaja en la empresa desarrollando la aplicación y realizando las pruebas necesarias.

Ingeniero de produccion y mantenimiento: (grado de ingeniería informática con especialidad en hardware) Provee de dar el soporte hardware y realizar el mantenimiento de este

Proveedor de BD: Empresa que proporciona los servidores de BD al sistema.

Administrador del sistema y Servicio al cliente: (FP en sistemas de gestión de bases de datos) Se encarga de administrar el sistema sobre todo en el apartado de las ubicaciones disponibles de la aplicación y proporciona el servicio de ayuda al cliente.

Usuarios: Hay 2 tipos:

- Usuarios de la app: Los usuarios que utilicen la aplicación. Se encargan de crear o inscribirse a encuentros deportivos con otros usuarios.

- Administrador: Es el administrador del sistema.

- Listado y descripcion de las inquietudes o intereses en el sistema

Se obtienen las siguientes inquietudes por las partes interesadas y algunos usuarios de prueba.

Inquietudes relacionadas con requisitos no funcionales:

Inquietudes generales:

Lo más importante es dar una buena sensación que den ganas de volver a usar la aplicación, esto se desglosa en 3:

- La aplicación no debe de ser muy complicada de procesar, para así poder usarse en cualquier dispositivo, por muy poca potencia que tenga, teniendo así muchos más posibles usuarios independientemente de los dispositivos que tengan.

- La aplicación debe de ser super rápida, para que no estropee la experiencia de el usuario por cargas lentas de algunos aspectos de la aplicación. También intentar optimizar las respuestas intentando que la velocidad de internet que se tenga no afecte mucho.

- Que sea fácil de usar para cualquier usuario, sea de la edad que sea, con una interfaz limpia e intuitiva y que puede adaptar el tamaño de letra para quien lo necesite.

Inquietudes específicas

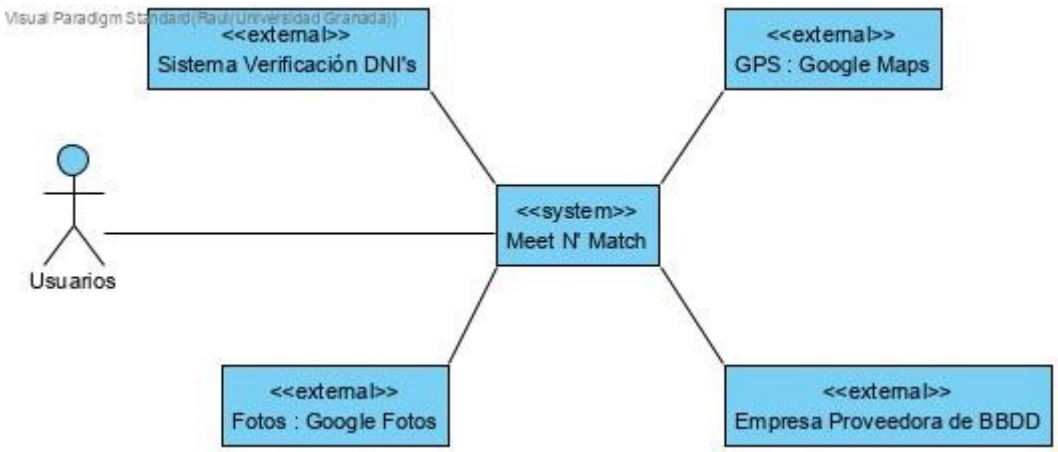
Responsivo : Que se adapte al dispositivo que lo usa, considerando los de menores potencia, para así reducir los procesos que debe ejecutar y sigan manteniendo la fluidez de los procesos de la aplicación.

Inquietudes relacionadas con requisitos funcionales:

Se añade un requisito funcional para atender la inquietud de un usuario, que decía que le gustaría que los usuarios los cuáles van a su encuentro, estuvieran verificados en la aplicación para saber que no es un “troleo” de que van a venir, y así no gastar recursos(dinero en gasolina,en billete de bus, etc) para luego no encontrarse a nadie allí. Por lo tanto los usuarios no verificados, no tienen acceso de primeras a todas las ubicaciones disponibles de la aplicación, solo a unas pocas.

Detalle	Descripción
RF#	10
Nombre	Verificar Usuario
Descripción	Verifica a un usuario en la aplicación permitiendole usar todas las ubicaciones disponibles.
Entrada	Dni del usuario y nickname
Procesamiento	[Prerrequisito: acceso identificado] Verificación de la cuenta en la BD externa al sistema, ; [Postrrequisito: acceso identificado verificado]
Salida	Confirmación de la verificación.

- **(0,5 puntos)** Diseña el modelo de contexto usando UML y un escenario funcional



Desarrollo de Software – Curso 2020-2021 - Examen primer parcial 9 abril 2021

ESCENARIO FUNCIONAL

- **Descripción general:** un usaurio se inscribe a un encuentro

- **Estado del sistema :** el usuario se ha identificado en el sistema

- **Entorno del sistema:** el entorno de implementacion funciona normalmente, sin problemas

- **Estmulo externo:** click sobre botón "Inscribirse a encuentro”

- **Respuesta requerida del sistema:** El sistema conecta con la base de datos externa donde estan almacenados los encuentros, muestra la lista de encuentros disponibles.

- **Estmulo externo:** click sobre el encuentro al cuál se quiere inscribir.

- **Respuesta requerida del sistema:** Se muestra mensaje de inscripción realizada correctamente y se envía por email al usuario una notificación, solo si tiene la opción de notificarme por email activada.

- **(2,5 puntos)** Elabora la descripción detallada de la vista o perspectiva que te corresponda según la tabla.

Descripcion detallada del punto de vista funcional

1 . Inquietudes

1.1 Capacidades Funcionales

- Alta Usuario
- Baja Usuario
- Consultar Datos Usuario
- Alta Ubicacion
- Baja Ubicacion
- Consultar Datos Ubicacion
- Crear Encuentro
- Mostrar Lista Encuentros
- Inscribir Usuario a Encuentro
- Verificar Usuario

Funciones que NO son responsabilidad del sistema:

- Cancelar/modicar encuentro (aún no)
- Pagar por otros medios el alquiler de una ubicación (aun no)
- Registrar transporte para ir al encuentro (aún no)

1.2 Interfaces Externas

- Consulta tipo SQL en BD externa
- Consulta de la localización de una ubicación (lleva a Google Maps)
- Consulta de fotos de una ubicación (lleva a Google Fotos)

1.3 Estructura Interna

Se ha asumido un estilo arquitéctónico general del tipo **“Cliente/Servidor”**. El servidor, en su mayoría está implementado **Kotlin** , usando **JVM**. Por el lado del cliente se usa **JavaScript** tambíen, pero la mayoría esta también implementado en **Kotlin**. Para las interfaces de la aplicación, se han usado **HTML5 y CSS3.**

Para mantener la maxima cohesion, se usara un mismo modulo para encapsular todo el software necesario de cada capacidad funcional completa (en el servidor de aplicaciones).

Para mantener un minimo acoplamiento, se uniran en el mismo modulo funciones que tengan una fuerte interaccion. Se usa la siguiente tabla 2x2 para representar la interaccion entre las distintas funciones, de forma que a partir de ella se decidira el número de modulos.

	Alta usuario	Baja usuario	Cons Datos usuario	Alta Ubi	Baja Ubi	Cons Datos Ubi	Crear Encuent	Mostrar Lista Encuent	Inscribir usuario Encuent	Verificar usuario
Alta usuario	**X**	**X**	**X**							**X**
Baja usuario	**X**	**X**	**X**							**X**
Cons Datos usuario	**X**	**X**	**X**							**X**
Alta Ubi				**X**	**X**	**X**				
Baja Ubi				**X**	**X**	**X**				
Cons Datos Ubi				**X**	**X**	**X**				
Crear Encuent							**X**	**X**	**X**	
Mostrar Lista Encuent							**X**	**X**	**X**	
Inscribir usuario Encuent							**X**	**X**	**X**	
Verificar usuario	**X**	**X**	**X**							**X**

Según las interdependencias que se observan en la tabla de arriba, creamos los siguientes módulos uniendo las distintas funcionalidades:

- M1 : Gestión Usuarios
  - Alta Usuario
  - Baja Usuario
  - Consultar Datos Usuarios
  - Verificar usuarios

- M2 : Gestión Ubicaciones
  - Alta Ubicación
  - Baja Ubicación
  - Consultar Datos Ubicación

- M3 : Gestión Encuentros
  - Crear Encuentro
  - Mostrar Lista Encuentros
  - Inscribir Usuario a Encuentro

1.4 Filosofía del diseño funcional

Se da importancia a otros principios de diseño:

- Coherencia: de forma que se ha hecho una descomposición en elementos correcta, para que interaccionen formando un todo.

- Cohesión: con todas las funciones relacionadas incluidas en un mismo elemento para realizar diseños menos proclives a errores.

- Consistencia: en cuanto a que los mecanismos y decisiones de diseño se aplican a través de toda la arquitectura facilitando el diseño y el mantenimiento.

Con estos 3 atributos se facilita la flexibilidad funcional ,siendo mucho más fácil de hacerlo evolucionar en el futuro con nuevas funcionalidades.

1.5 Inquietudes de cada interesado en el sistema

Tipo de interesado	Inquietudes
Desarrollador y técnico de pruebas	Estructura interna y cualidades de diseño básicas; capacidades funcionales e interfaces externas.
Ingeniero de produccion y mantenimiento:	Estructura interna y cualidades de dise~no basicas, y mantenimiento interfaces externas y capacidades funcionales.
Proveedor de BD:	Capacidades funcionales basicas, interfaces externas, en especial las que conectan con otros elementos internos a la empresa.

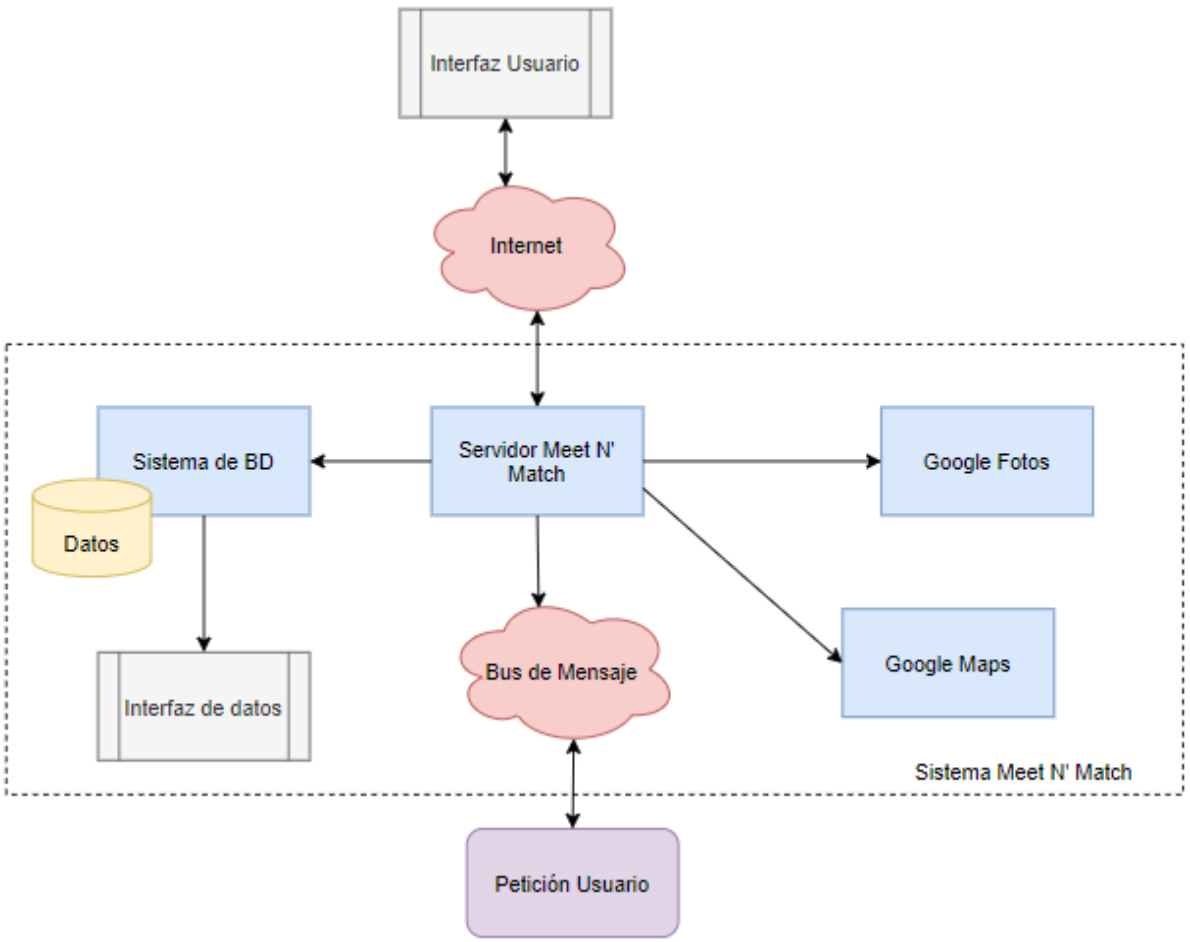
Administrador del sistema y Servicio al cliente:	Capacidades funcionales básicas, filosfía de diseño funcional básica, gestión de funciones configurables, interfaces externas y estrctura interna.
Usuarios:	Capacidades funcionales básicas e interfaces externas.

2. Modelos

2.1 Modelo de la estructura funcional

Utilizaremos un modelo que tenga en cuenta los siguientes componentes:

- Elementos funcionales
- Interfaces (de los elementos funcionales del sistema)
- Conectores (entre los elementos funcionales del sistema)



3. Actividades

3.1 Identificar los elementos

Para elaborar el diagrama de clases, hemos partido de las capacidades funcionales y agrupado las mismas identificando los conceptos más importantes para deducir a partir de éstos las entidades (clases). Hemos aplicado sobre todo la composición para agrupar clases en paquetes. Otras tareas de refinamiento se dejarán para la fase de diseño.

3.2 Asignar responsabilidades a los elementos

A cada clase se le han asignado las capacidades que están dentro de un mismo módulo, según la estructura interna.

- Elemento: Aplicación Meet N' Match
  - Proporcionar a los clientes la interfaz accesible en la app.
  - Gestionar todos los estados relacionados con la sesión de la interfaz del usuarios.
  - Interaccionar con otras partes del sistema para permitir que el usuario vea la lista de encuentros disponibles y consultar su propia información.

- Elemento: Gestión de Usuarios
  - Gestiona toda la información persistente relacionada con los usuarios.
  - Registra cada transacción en la BD.

- Elemento: Gestión de Ubicaciones
  - Gestiona toda la información persistente relacionada con las ubicaciones.
  - Registra cada transacción en la BD.

- Elemento: Gestión de Encuentros
  - Gestiona toda la información persistente relacionada con los encuentros.
  - Registra cada transacción en la BD.

- Elemento: Interfaz para gestión de ubicaciones
  - Permite que el administrador del sistema pueda acceder al elemento "Gestión de Ubicaciones" para administrar las ubicaciones
  -
- Elemento: Interfaz con Google Maps
  - Proporciona la conexión entre el servidor y el servicio GPS de terceros Google Maps
  - Permite hacer operaciones de consulta de indicaciones a este servicio
  -
- Elemento: Interfaz con Google Fotos
  - Proporciona la conexión entre el servidor y el servicio Visualizador de Imágenes de terceros Google Fotos
  - Permite hacer operaciones de consulta de indicaciones a este servicio

3.3 Diseñar las interfaces

Como la conexión no es síncrona por RPC sino basada en mensajería (REST), se usará para describirlas un enfoque orientado a datos.

Detalle	Descripción
Nombre	altaUsuario
Semántica	Se envían datos de usuario y se registra el alta
Tipo	REST(mensajería): PUT
Prerequisitos	No estar identificado como usuario
Entrada	nickname , email , contraseña, nombre, edad, teléfono
Salida	confirmación alta

Detalle	Descripción
Nombre	verificarUsuario/ bajaUsuario
Semántica	se envían datos para verificar
Tipo	REST(mensajería): PUT
Prerequisitos	Estar identificado como usuario
Entrada	DNI si es para verificar o nada si es para darse de baja
Salida	Confirmación de verificación o baja;vuelta al acceso no identificado

Detalle	Descripción
Nombre	crearEncuentro
Semántica	se envía la información para crear un encuentro
Tipo	REST(mensajería): PUT
Prerequisitos	Estar identificado como usuario
Entrada	Nickname, ubicación, hora
Salida	Confirmación de creación de encuentro

Detalle	Descripción
Nombre	consultaUbicaciones
Semántica	Se muestran las ubicaciones
Tipo	REST (mensajería): GET
Prerequisitos	Estar identificado
Entrada	-
Salida	Listado de las ubicaciones

Detalle	Descripción
Nombre	altaUbicación/bajaUbicación
Semántica	Añadir o borrar ubicaciones
Tipo	REST(mensajería): PUT
Prerequisitos	Estar identificado como administrador
Entrada	Datos ubicación o el nombre de la ubicación que queramos dar de baja
Salida	Confirmación del alta o de la baja.

Detalle	Descripción
Nombre	mostrarEncuentros
Semántica	Se muestran los encuentros
Tipo	REST(mensajería): GET
Prerequisitos	Estar identificado
Entrada	-
Salida	Listado de los encuentros

Detalle	Descripción
Nombre	inscripciónEncuentro
Semántica	Se envían los datos del usuario para inscribirse al encuentro
Tipo	REST(mensajería): PUT
Prerequisitos	Estar identificado como usuario
Entrada	Nickname del usuario y datos del encuentro
Salida	Confirmación de la inscripción

3.4 Diseñar los conectores

Todos los conectores entre cliente y servidor serán por mensajería HTTP (servicios REST), usarán páginas java del lado del servidor (JSP) cuando los datos se puedan pasar directamente a la BD , cuando se requiera cierto procesamiento. Del lado del cliente, utilizaremos consultas a través de AJAX haciendo uso de Jquery.

3.5 Comprobar la trazabilidad funcional

Se puede consultar en la tabla de trazabilidad y la estructura interna elaborada a partir de ella en la sección **“1.3 Estructura Interna” .**

3.6 Recorrer escenarios comunes

Por parte del desarrollador y tecnico de pruebas debe revisar los escenarios de interacción entre cliente y servidor , pero esta tarea queda relegada a la fase de diseño.

3.7 Analizar las interacciones

Como se ha aplicado en enfoque orientado a objetos en las conexiones entre cliente y servidor,se emulan las conexiones entre conceptos del mundo real, reduciendo de forma natural las interacciones entre conceptos (bajo acoplamiento) y aumentando las internas a un mismo concepto (alta cohesión).

3.8 Analizar la flexibilidad

La estructura OO favorece la flexibilidad de todo el sistema, tanto para añadir como modificar módulos de software y facilita el mantenimiento junto a la evolución del sistema.

4. Problemas y errores comunes

- Interfaces pobremente definidas: -
- Responsabilidades no bien entendidas: -
- Vista sobrecargada: -
- Diagramas sin definiciones de elementos: -
- Dificultades para reconciliar las necesidades de distintas partes interesadas: -
- Nivel de detalle erróneo: -
- “Elementos divinos”: -
- Demasiadas dependencias: -

5. Lista de Verificación

¿Tiene menos de 15 a 20 elementos de nivel superior? Sí

¿Tienen todos los elementos un nombre, responsabilidades claras e interfaces claramente definidas? Sí

¿Tienen lugar todas las interacciones de elementos a través de interfaces y conectores bien definidos que unan las interfaces? Sí , aunque se describen de forma textual

¿Exhiben los elementos un nivel apropiado de cohesión? Sí

¿Exhiben los elementos un nivel apropiado de acoplamiento? Sí

¿Ha identificado los escenarios de uso importantes y los ha utilizado para validar la estructura funcional del sistema? No, se considera que los que restan es mejor dejarlo para la fase de diseño.

¿Ha verificado la cobertura funcional de su arquitectura para asegurarse de que cumple con los requisitos funcionales? Sí

¿Ha definido y documentado un conjunto apropiado de principios de diseño arquitectónico y su arquitectura cumple con estos principios? Sí

¿Ha considerado cómo es probable que la arquitectura haga frente a posibles escenarios de cambio en el futuro? Sí

¿La presentación de la vista tiene en cuenta las preocupaciones y capacidades de todos los grupos de partes interesadas? ¿Actuará la vista como un vehículo de comunicación eficaz para todos estos grupos? Sí, se han realizado tres modelos distintos para logarlo

4. **Pruebas software (1 punto):** A partir de la parte de la DA realizada en el ejercicio 3:
   - **(0,5 puntos)** Realiza una planificación de las pruebas[4] que tenga en cuenta solo los requisitos que te correspondan según la tabla.
   - **(0,5)** A partir de la descripción detallada de la vista o perspectiva que hayas realizado, especifica las condiciones a probar (análisis de pruebas).

4. En concreto debes responder a (1) Qué se quiere probar y cómo, (2) quién lo hará y cuándo, y (3) qué criterios se usarán para decidir cuándo terminar las pruebas.