Juego multijugador web de simulación social

Mariano López Muñoz

Junio de 2013

Índice

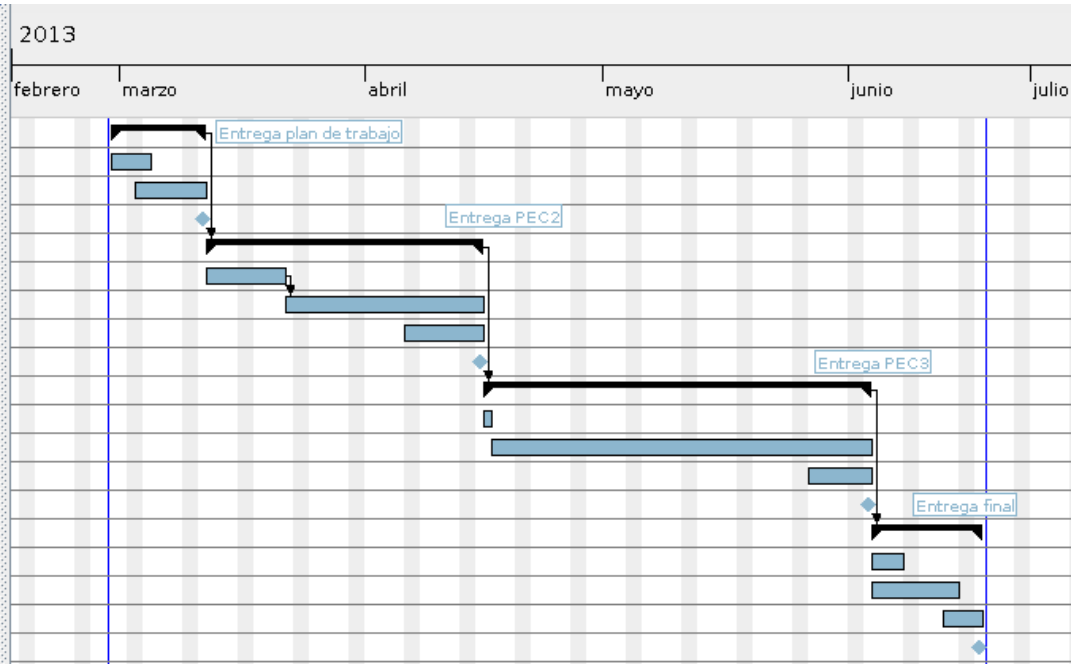
Crear un juego multijugador web

- Sencillo por fuera, "jugable" por dentro.
- Libre, y utilizando solamente herramientas libres.
- Para profundizar en las tecnologías:
  - Java EE 6.
  - JavaServer Faces 2.

Cumplir la planificación

- Tres entregas parciales y una final.

Nombre	Fecha de fin
Plan de trabajo	11/03/13
Planteamiento y resolución de dudas	4/03/13
Elaboración plan de trabajo	11/03/13
Entrega plan de trabajo	11/03/13
PEC2	15/04/13
Análisis	21/03/13
Diseño	15/04/13
Inicio de la memoria	15/04/13
Entrega PEC2	15/04/13
PEC3	3/06/13
Preparación del entorno de trabajo	16/04/13
Implementación	3/06/13
Avance de la memoria	3/06/13
Entrega PEC3	3/06/13
Memoria y presentación	17/06/13
Revisión y empaquetado software	7/06/13
Finalización de la memoria	14/06/13
Presentación virtual	17/06/13
Entrega final	17/06/13



Herramientas de edición

- NetBeans ⇒ Entorno de desarrollo integrado.
- Lyx ⇒ Memoria y presentación.
- ArgoUML ⇒ Gráficos UML.
- GanttProject ⇒ Gráfico Gantt de la planificación.
- Gimp ⇒ Manipulación de imágenes.

Bibliotecas y marcos de trabajo

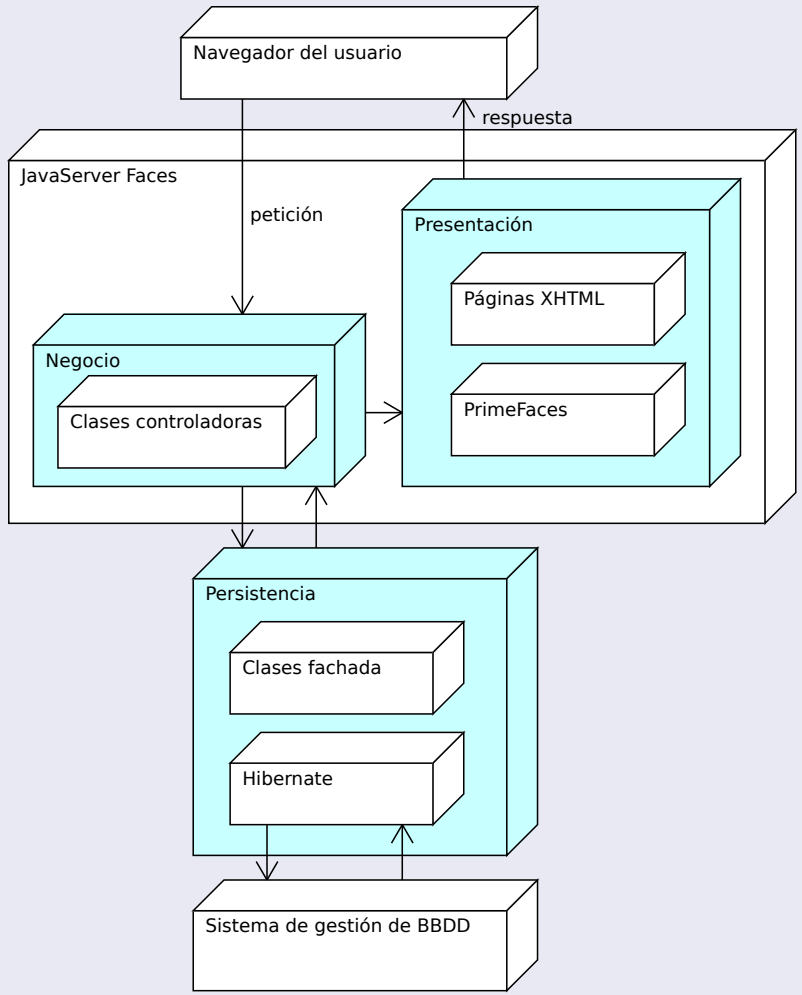
- JavaEE 6
  - Enterprise JavaBeans (EJB) 3.1
  - Java Persistence API (JPA) 2.0
  - ...
- JavaServer Faces 2 + PrimeFaces
- Hibernate
- PostgreSQL
- JBoss Application Server 7



Arquitectura lógica

Las tres capas clásicas:

- Presentación
  - Páginas XHTML
  - PrimeFaces
- Negocio
  - Clases controladoras
- Persistencia
  - Clases fachada
  - Hibernate



Enterprise JavaBeans 3.1

- Especifica los servicios que el servidor de aplicaciones nos proporciona.
- Usa extensivamente *annotations*.
- Nos permite crear fácilmente métodos asíncronos y métodos que se lancen periódicamente.

Ejemplo

```
@Schedule(hour = "*/2", minute = "0", second = "0")
public void pasaTurno(Timer timer) {
  ...
}

@Asynchronous
public Future<Boolean> procesa(Accion accion) {
  boolean resultado;
  ...
  return new AsyncResult<Boolean>(resultado);
}
```

Java Persistence API 2.0

- Nos sirve para definir la capa de persistencia.
- También usa extensivamente *annotations*.

Ejemplo

```
@Entity
public class Personaje {
  ...
  @ManyToOne(optional = false)
  private Localizacion localizacion;

  @OneToMany(mappedBy = "personaje")
  private Set<Elemento> elementos;

  @OneToOne(optional = false)
  private ConjuntoHabilidades conjuntoHabilidades;
  ...
```

Localización

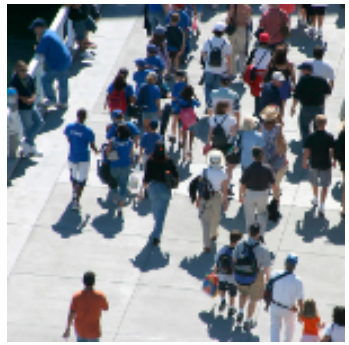
Cada uno de los lugares donde pueden estar los personajes:

- Le corresponden unas determinadas coordenadas.
- Está comunicado directamente con otras localizaciones.
- Proporciona una serie de recursos naturales.

Personaje

¡El protagonista de la historia!

- Está en una localización y puede viajar a otras.
- Interactúa con otros personajes y elementos.
  - Hablando o susurrando.
  - Cogiendo, soltando o dando elementos.
  - Creando o participando en trabajos.
  - Escribiendo notas.
  - Atacando.
- Cada jugador puede manejar varios.



Elemento

Cualquier cosa que un personaje pueda llevar encima:

- Alimento, medicina o veneno.
- Arma de ataque o defensa.
- Material.
- Herramienta.
- Recurso natural.
- Nota.

Ejemplo

Creamos el tipo “martillo”, que sirve como herramienta y como arma.

Trabajo

Proceso para crear elementos nuevos:

- Tarda uno o más turnos en completarse.
- De la "nada": extrayendo un recurso local.
- Construyendo: según un tipo de trabajo que definamos.
  - Consumiremos materiales.
  - Necesitaremos herramientas.
  - Obtendremos una salida.

Ejemplo

$$\text{Construir silla: } \overbrace{4\,\text{patas} + 1\,\text{asiento} \rightarrow 1\,\text{silla}}^{\text{Dispongo de 1 martillo}}$$

Habilidades

Cada personaje es bueno con algunos trabajos... y no tan bueno con otros.

- Para un personaje, representa sus capacidades.
- Para un trabajo, representa su dificultad.
- Se componen de cuatro parámetros:
  - Fuerza
  - Agilidad
  - Destreza
  - Percepción
- Su existencia fomenta la colaboración entre distintos personajes con habilidades complementarias.

Dinámica

La dinámica del juego funciona a dos niveles:

- Inmediato: En cuanto el personaje realiza la acción, el sistema la gestiona y ofrece un resultado.
  - Hablar, mover elementos, crear o leer notas, atacar...
- Por turnos: Hay que esperar al siguiente turno para ver el resultado.
  - Avanzar en un trabajo o terminarlo.
  - Viajar hacia otra localización.

Módulos y actores

- Gestión de localizaciones ⇒ Editor de mapas.
- Gestión de tipos de elemento ⇒ Editor de tipos.
- Gestión de tipos de trabajo ⇒ Editor de tipos.
- Gestión del personaje ⇒ Jugador.
- Motor del sistema ⇒ Reloj del sistema.

Resultado y lecciones aprendidas

- Java EE 6 es un estándar muy maduro y potente.
  - Podemos implementar mucha funcionalidad escribiendo poco código.
- Podemos realizar todas las fases de desarrollo software con herramientas libres.
  - Desde el sistema operativo GNU/Linux.
  - Pasando por cada una de las tecnologías y marcos de trabajo utilizados.
  - Hasta la última herramienta de edición.
- Mucho margen para mejoras y añadir funcionalidades.
- El código fuente ya está publicado bajo licencia GPL 3.
  - ¡Ahora sólo queda echarle ganas e imaginación!

Continuará...

¡Gracias!

URL: http://github.com/dertalai/triican

Contacto: malomu@uoc.edu