# Introducción

## Ámbito del proyecto y motivación

## Objetivos generales del proyecto

## Objetivos específicos

## Respecto al diseño de la aplicación

## Estructura de la memoria

# Tecnologías

## Node.js

Node.js es un entorno de programación en la capa del servidor basado en JavaScript, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Es una idea completamente revolucionaria ya que, a diferencia de las aplicaciones web tradicionales que usan PHP en el servidor, utiliza JavaScript, un modelo completamente asíncrono.

Uno de los principales motivos por el cual he escogido Node.js, es porque es especialmente útil cuando se van a realizar muchas operaciones simultáneas, sobretodo operaciones I/O. También es realmente bueno para aplicaciones *realtime*, ya que necesitan una conexión persistente entre el cliente y el servidor.

Además, consta de un gestor de paquetes (NPM o *Node Package Manager*), que facilita la tarea de incluir librerías en nuestro proyecto. Todas las librerías utilizadas en mi proyecto (y que explicaré a continuación) han sido incluidas utilizando esta utilidad.

### Express

Express es un “framework de desarrollo de aplicaciones web minimalista y flexible para Node.js”. Ofrece, entre otras características, un *router* de URL (*get*, *post*, *put*), el cual vamos a usar para capturar los diferentes eventos que se puedan producir, desde cambiar de página, hasta hacer operaciones con la base de datos.

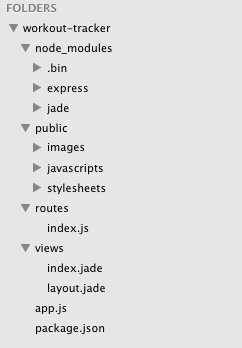
Para instalarlo, usaremos NPM:

npm install -g express

Además, Express nos ofrece la posibilidad (y recomendación) de crear un proyecto predefinido, con las funcionalidades básicas y a partir del cual empezar a desarrollar nuestra aplicación siguiendo los fundamentos del *framework*. Para crearlo, solo tenemos que ejecutar la siguiente línea de código:

express ./nombre\_proyecto

Esto nos creará una estructura de ficheros similar a la siguiente:



* **node\_modules:** Contiene las dependencias instaladas mediante NPM.
* **public:** Contiene todos los recursos que estarán disponibles para la parte del cliente, tales como imágenes, ficheros JavaScript, u hojas de estilo (CSS).
* **routes:** Aquí se encontrarán los métodos que se ejecutarán cuando se capte un evento mediante *get* o *post*. Se puede decir, que es como el intermediario entre la petición del cliente y la ejecución de la misma.
* **views:** Contiene la vista del proyecto, las plantillas que generarán nuestra página web. En este proyecto en vez de utilizar HTML vamos a usar JADE (explicado más adelante), para poder generar páginas dinámicas.
* **app.js:** El fichero principal de la aplicación, el cual se ejecutará para arrancarla.
* **package.json:** Este fichero cinluye las declaraciones de las dependencias de nuestra aplicación.

Un ejemplo de la ruta que seguiría una petición desde el cliente hasta el servidor sería el siguiente:

/app.js -> /routes/game.js -> /db/upgrades.js

En cuanto a la creación del servidor para nuestra aplicación web, Express también nos facilita la faena. De todos modos, nosotros hemos creado el servidor utilizando WebSockets, de manera que

## MongoDB

## WebSockets

## Jade

Jade es un lenguaje de plantillas desarrollado por el creador de Express para simplificar la sintaxis de HTML y acelerar el proceso de desarrollo.

Este lenguaje intercambia tener que cerrar etiquetas HTML por la indentación, es decir, todo bloque de texto que esté hacia la derecha de la etiqueta que abre, significa que va dentro.

También elimina los símbolos “<” y “>”, y los parámetros de las etiquetas se pasan entre paréntesis como si fueran de una función de cualquier lenguaje de programación.

Un ejemplo comparativo entre HTML y Jade:

h1(id="title") Welcome to Jade

button(class="btn", data-action="bea").

Be Awesome

Equivaldría a:

<h1 id="title">Welcome to Jade</h1>

<button data-action="bea" class="btn">Be Awesome</button>

Para añadir clases o IDs a los diferentes elementos, podemos utilizar el “.” o la “#” en vez de pasarlo parámetro. Por ejemplo:

table.class1

table#id1

Equivale a:

table(class=”class1”)

table(id=”id1”)

También cabe mencionar, que si el elemento en cuestión es un *div*, se puede omitir. Por lo tanto:

.class1

Equivale a:

div.class1

Otra de las funcionalidades fundamentales por la cual he escogido Jade, es que éste soporta JavaScript incrustado en el documento. De este modo, podemos añadir variables corrientes, listas, e incluso iterar con la sentencia *foreach*.

## Links:

http://mongoosejs.com/docs/