

Sergio Cobos García

Yura Turlay

David Luna Fernández

Proyecto CFGS 2º Desarrollo de Aplicaciones Web

**Índice**

Introducción………………………………………………………………………………………………………………………Pag.3

Tecnologías usadas……………………………….…………………………………………………………………………..Pag.7

Diseño de la aplicación web……..………………………………………………………………………………………Pag.15

Diseño de la Base de datos……………………………………………………………………………………………… Pag.19

Desarrollo y despliegue……………………………………………………………………………………………………Pag.24

Bibliografía…………………………………….…………………………………………………………………………………Pag.36

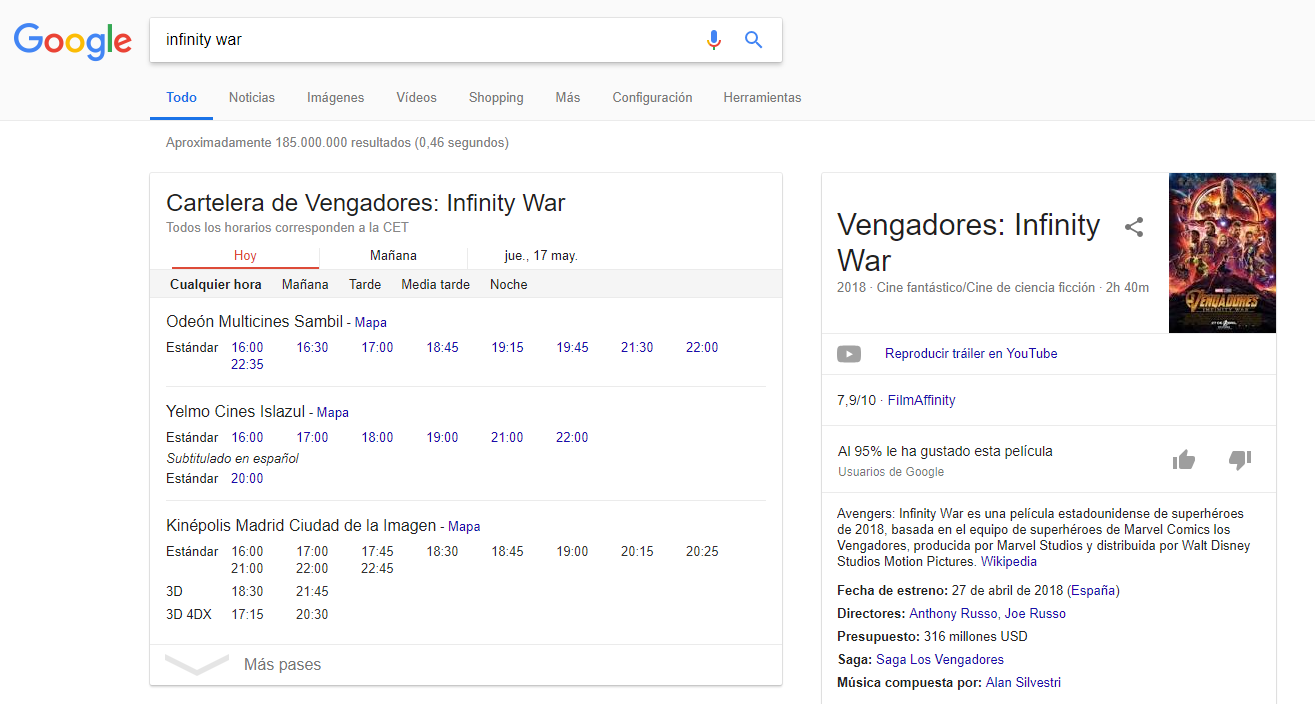
**Introducción**

YoSoyTuCine es un buscador/comparador de cines, no de películas.

Entender esta idea es básica para comprender el proyecto que abordamos ya que comparadores y buscadores de películas hay muchos e imposible de competir con ellos debido a su gran comunidad de usuarios, véase FilmAffinity.

La idea del proyecto es poder tener en simple vistazo, gracias a la geolocalización, tu cine más cercano.

Por ejemplo, Google, a fecha de entrega de este proyecto, si buscas una película, la información que te propone son las sesiones y cines disponibles, pero no tu cine más cercano.

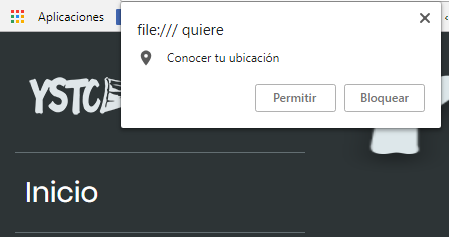


La idea de YoSoyTuCine es además ordenar las películas primeramente por cercanía a tu ubicación en todo momento, y con posibilidad a hacerlo en base a las valoraciones dadas a los cines por la comunidad de usuarios de YoSoyTuCine.

Otro punto a destacar de YoSoyTuCine es que evidentemente al no poder competir con buscadores omnipotentes como Google, intentamos ofrecer la información buscada por el usuario en un menor número de clics, esa es la ventaja que proponemos, ya que simplemente al entrar en la web, serás preguntado por tu ubicación y tendrás ante tus ojos los carteles de las películas en cartelera actualmente.

Pinchando en el cartel tendrás todos los cines a tu alrededor ordenados por distancia y la información de las sesiones, de los cines y de la película en la que acabas de hacer clic.

Para lograr los objetivos finales de YoSoyTuCine hemos de usar las funciones de geolocalización de Google para poder establecer nuestra ubicación,





y poder hacer los cálculos de distancia por latitud y longitud entre nuestra ubicación actual y la del cine, la cual es almacenada en base de datos y que es rellenada automáticamente tanto en cines como en películas que están en cartelera mediante la técnica de scrapping.

La ubicación proporcionada por la API de google la almacenamos en una cookie que dura lo que dura la sesión del usuario en nuestra web, de tal modo que no tenga que estar siendo preguntado por su ubicación mientras navega por la web, y que si sea preguntado al volver a entrar por si ha cambiado su ubicación (al consultar YoSoyTuCine desde un smartphone por ejemplo).

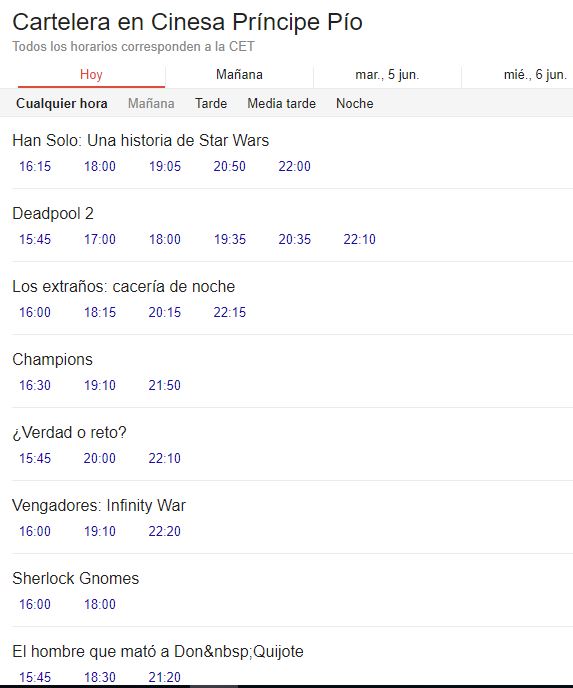
La técnica de scrapping mediante la cual rellenamos la base de datos, es básicamente obtener información de una web, aquí vamos a ver las definiciones:

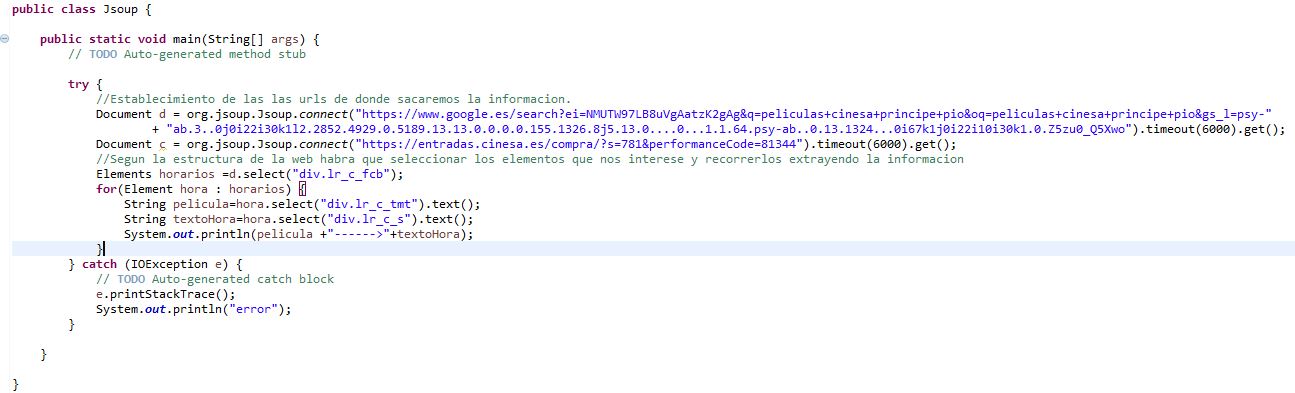
*Web scraping es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.*

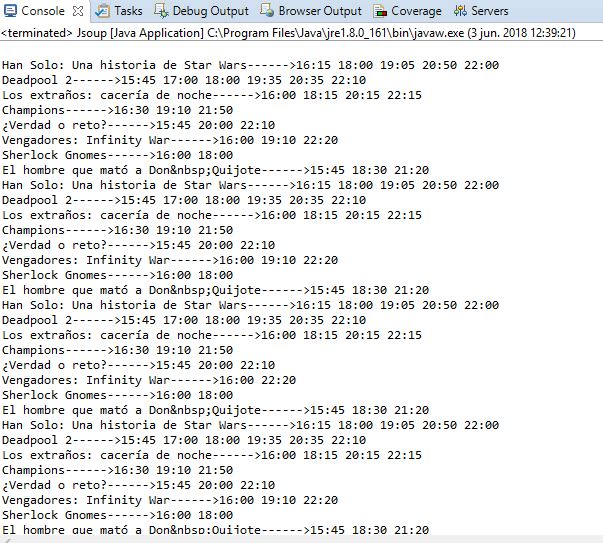
*El web scraping está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el web scraping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados en una base de datos central, en una hoja de cálculo o en alguna otra fuente de almacenamiento. Alguno de los usos del web scraping son la comparación de precios en tiendas, la monitorización de datos relacionados con el clima de cierta región, la detección de cambios en sitios webs y la integración de datos en sitios webs.*

En nuestro caso el uso del scrapping es para obtener de las paginas como Cinesa la información de una película y poder rellenar nuestra base de datos con la información obtenida, por ejemplo lanzando el scrapping cada viernes podemos actualizar nuestra base de datos con las películas nuevas o borrar las películas que ya no tenemos en cartelera.

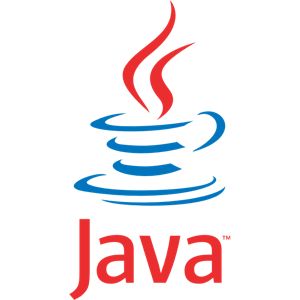
Aquí podemos ver un ejemplo de código de scrapping donde recuperamos información de la cartelera:

******



Aquí vemos el resultado y como hemos conseguido recuperar la información de dicha web.

**Tecnologías usadas**

****

****

****

****

****

****

**JavaScript**

Utilizado para generar el dinamismo de la web en conjunción con JQuery.

**Ajax**

Utilizado para evitar las recargas de la web a la hora de filtrar en las búsquedas.

**Java**

Utilizado bajo el patrón MVC, es el encargado de dirigir la aplicación web, recibir peticiones, ir a base de datos y devolver vistas.

**MySQL**

MySQL ha sido el gestor de base de datos elegido debido a su facilidad de uso y a su ligereza que hacen de él la opción más utilizada para este tipo de proyectos Web.

**Bootstrap**

Librería de estilos creada y liberada por Twitter con una gran popularidad y aceptación en el mundo del diseño web gracias a las facilidades que aporta y que usamos en el proyecto como es su sistema de rejilla.

**HTML5**

Estándar, es el esqueleto de la aplicación web.

**CSS3**

Estándar en cuanto a estilos de una web.

**JQuery**

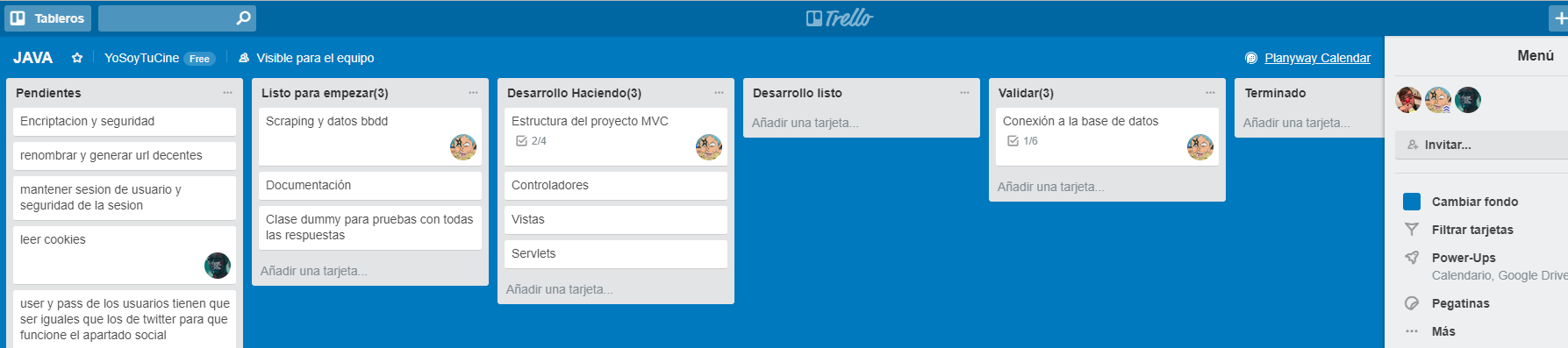
Encargado de generar HTML dinámicamente y elegido por su rapidez y sencillez en cuanto a código para realizar dicha tarea.

Todas estas tecnologías han sido abordadas en mayor o menor medida durante el ciclo y por tanto no es necesario desarrollarlas más en esta documentación, a continuación explicaremos más detenidamente dos de las cuales no se abordaron durante el curso.

**Trello**

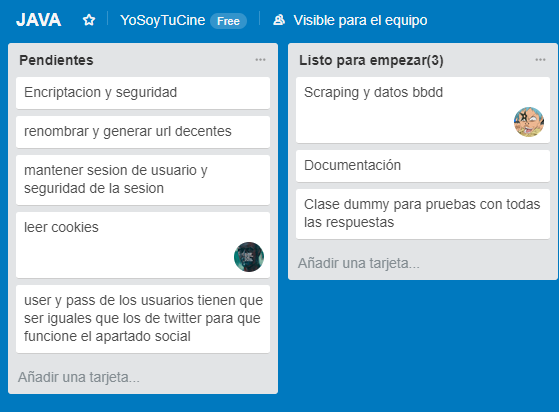
Trello es una aplicación Web la cual te proporciona un sistema de trabajo basado en metodologías agiles.

Lo que podemos ver aquí es un tablero Kanban en el cual se distribuyen las tareas entre los distintos miembros del equipo y durante las distintas fases del desarrollo de una tarea.



En nuestro caso hemos creado distintos tableros para diferenciar tecnologías, por ejemplo aquí vemos el tablero para las tareas que tienen que ver con JAVA.

A su vez disponemos de otros tableros creados para las tareas de Base de Datos, de HTML, CSS y JS… etc.



Los estados por los cuales pasa una tarea son los vistos en las capturas adjuntas:

Pendiente: Donde se ponen las tareas que necesitaremos, es una pila de las tareas que englobaran el proyecto.

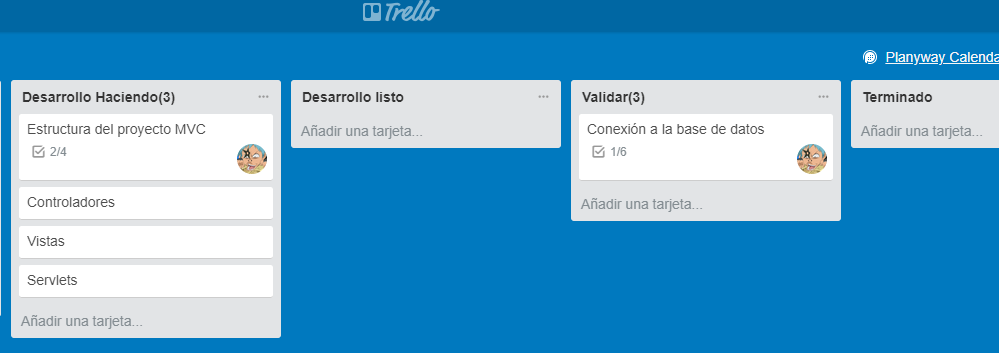
Listo para empezar: significa que dicha tarea ya tiene las dependencias necesarias completadas y está lista para ser desarrollada.

Haciendo: Tarea en desarrollo.

Listo: Es donde se colocan las tareas terminadas listas para que el encargado del equipo de la validación coja la tarea y se la lleve a su campo y pueda revisarla y decida si tiene que volver a desarrollo o está terminada.

Validar: Tarea en validación.

Terminado: Utilizado como guía visual del trabajo realizado hasta ahora.



Cada tarea puede tener subtareas, adjuntos, tags, fechas de entrega…



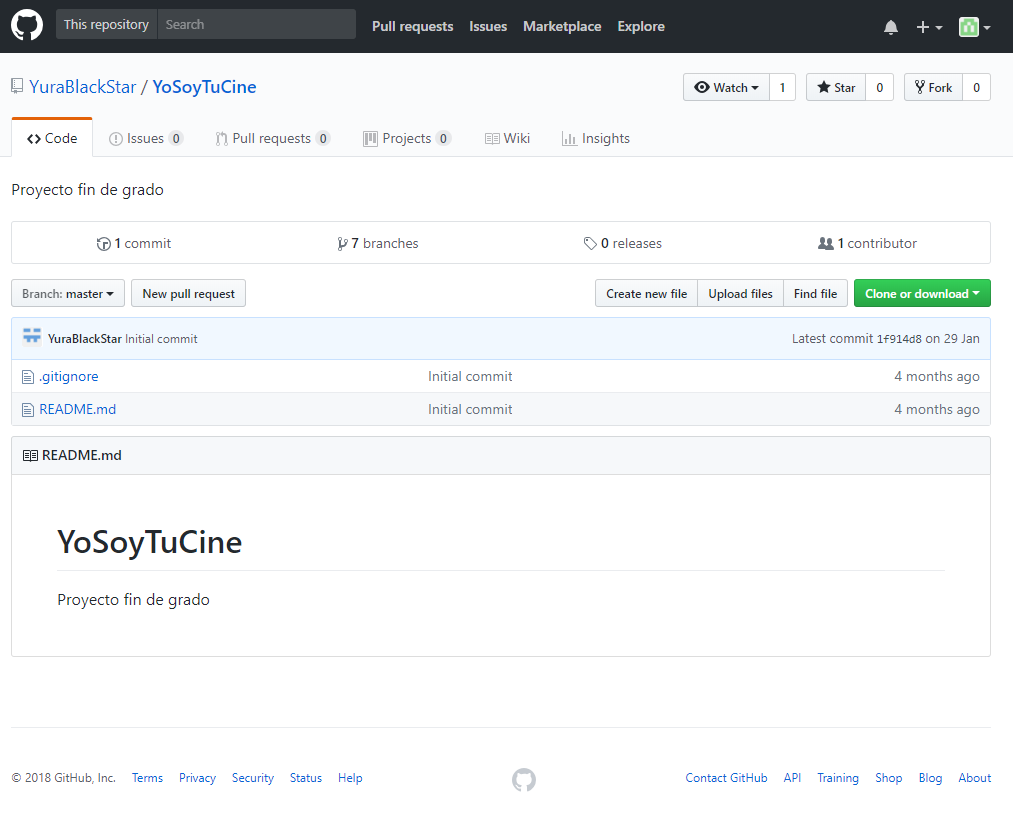
Como vemos Trello es una aplicación de gran utilidad para controlar y organizar el trabajo de un equipo de desarrollo, y es ampliamente usado en todo el mundo del desarrollo.

**Github**

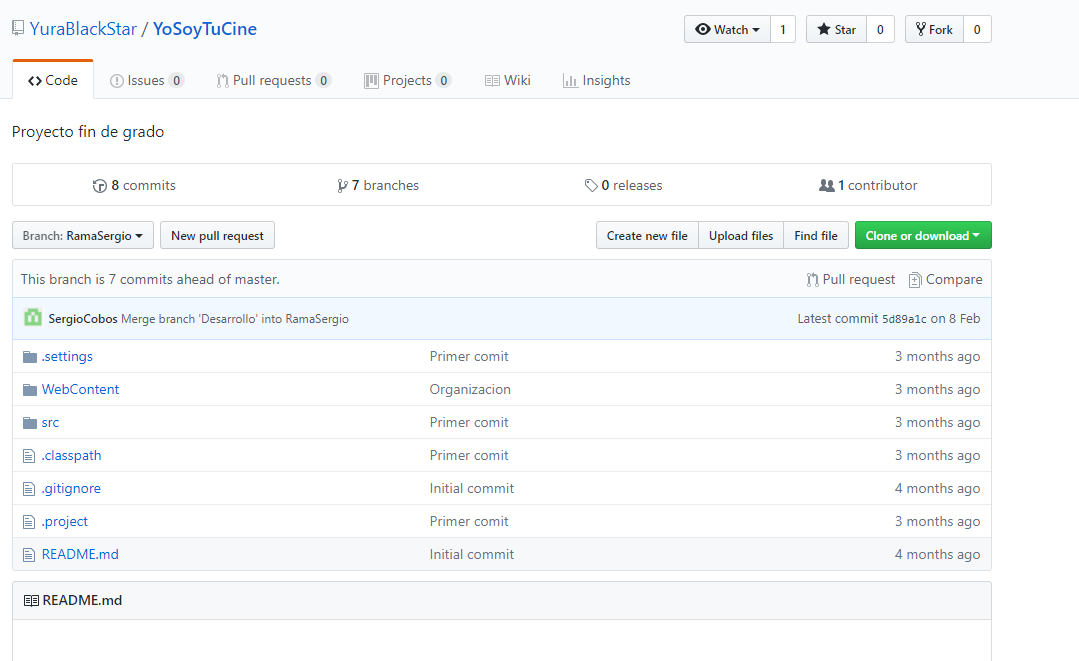
Github es la comunidad/red social de los programadores por excelencia y está basado en el software de control de versiones GIT, creado por Linus Torvalds.

Para nuestro proyecto hemos decido usar control de versiones para familiarizarnos con esta herramienta absolutamente imprescindible y fundamental en el mundo del desarrollo.

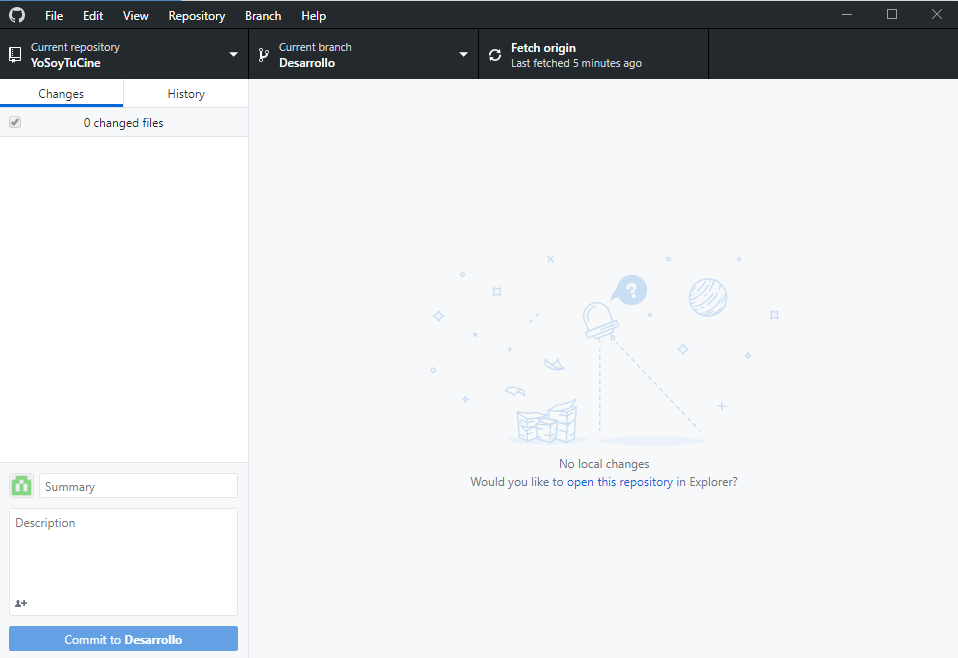
Creamos un repositorio para el proyecto en el cual los integrantes del equipo de desarrollo podemos programar simultáneamente al disponer siempre de las versiones actualizadas de los códigos, gracias a funcionalidades de los software de control de versiones como es la función Merge que te permite que varias personas trabajen sobre un mismo código integrando al finalizar de programar las líneas de todo el equipo.



La forma elegida para trabajar ha sido la de crear una rama master, una rama desarrollo y varias ramas individuales para trabajar cada uno sobre nuestra rama e ir integrándolas sobre la rama desarrollo y una vez esta testeado integrarlo en la master, el control de versiones te permite tener un absoluto control sobre las diferentes versiones del proyecto y te permite volver atrás y deshacer errores con gran facilidad y mantener un orden sin volverte loco.



Puedes trabajar mediante la consola de GIT y tener acceso a todas los comandos o para familiarizarte y aprender las funciones más básicas (en nuestro caso no tenemos que hacer uso de comandos avanzados) usar el cliente de escritorio, que es tremendamente sencillo y fácil de usar, a simple vista podemos ver cómo podemos elegir las ramas, descargarnos la última versión del código, hacer commit de los cambios, hacer merge de ramas, subir los cambios al repositorio remoto…etc.



**Diseño de la web**

Una vez teníamos la idea clara de la aplicación web que queríamos hacer tuvimos que realizar fue un análisis de requerimientos para saber cuántas paginas tendría nuestra web, que pasaría al pinchar en cada enlace, cuál iba a ser nuestra estructura, tecnologías que usaremos (AJAX o recargar la página, bootstrap u otro framework css...) Etc…

Una vez con toda esa información aclarada y acordada entre el equipo diseñamos los primeros bocetos con herramientas de prototipado tales como mockups generando los siguientes bocetos para las páginas que previamente habíamos acordado que tendría la página:

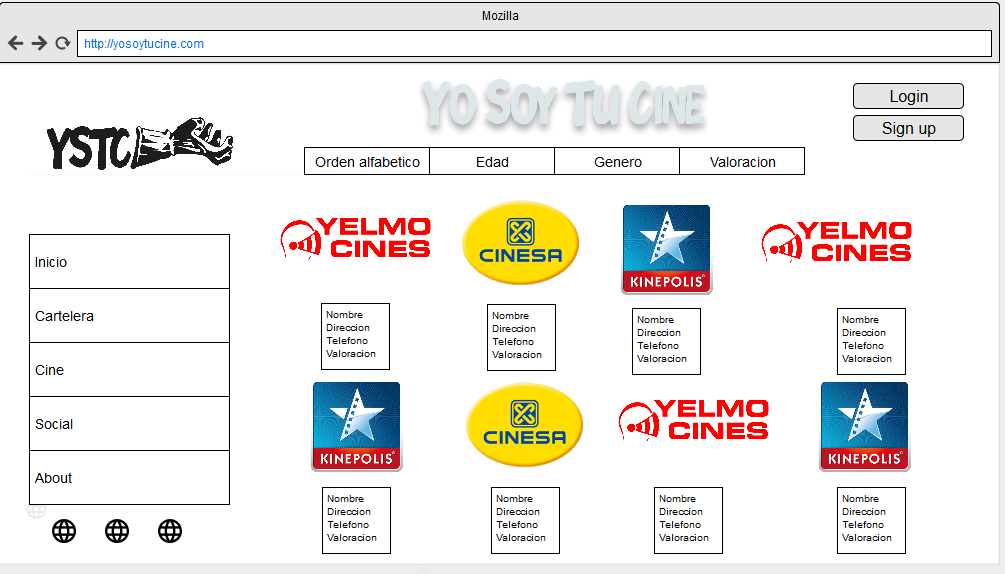
**Index.jsp**



**cartelera.jsp**



**cines.jsp**



Con los bocetos ya creados fuimos creando la estructura HTML y los estilos CSS

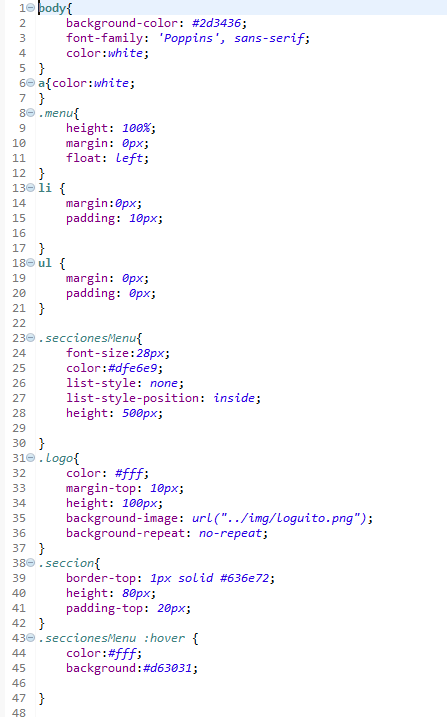
**HTML**



**USO DE BOOTSTRAP**



**CSS**



**Diseño de la base de datos**

Dentro del primer análisis, también fue crucial acordar el diseño de nuestra base de datos, para ello se analizó cual serían las entidades necesarias para nuestra web, realizando un primer modelo entidad relación que posteriormente, una vez acordado, pasamos a modelo relacional y que finalmente se creó físicamente con los siguientes scripts.

CREATE TABLE empresas

(idempresa INTEGER NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(150) NOT NULL,

logo VARCHAR(4000),

PRIMARY KEY(idempresa)

);

CREATE TABLE cines

(idcine INTEGER NOT NULL AUTO\_INCREMENT,

idempresa INTEGER NOT NULL,

nombre VARCHAR(150) NOT NULL,

latitud DOUBLE NOT NULL,

longitud DOUBLE NOT NULL,

direccion VARCHAR(300) NOT NULL,

cp INTEGER NOT NULL,

ciudad VARCHAR(150),

telefono INTEGER,

valoracion FLOAT,

url VARCHAR(4000),

PRIMARY KEY(idcine),

FOREIGN KEY(idempresa)

REFERENCES empresas(idempresa)

ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE sesiones(

fecha DATE NOT NULL,

hora VARCHAR(5) NOT NULL,

PRIMARY KEY(fecha,hora)

);

CREATE TABLE cinesysesiones(

idcine INTEGER NOT NULL,

fecha DATE NOT NULL,

hora VARCHAR(5) NOT NULL,

precio INTEGER,

PRIMARY KEY(idcine,hora,fecha),

FOREIGN KEY(idcine)

REFERENCES cines(idcine)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(fecha,hora)

REFERENCES sesiones(fecha,hora)

ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE peliculas(

idpelicula INTEGER NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(150) NOT NULL,

genero VARCHAR(150) NOT NULL,

edadrecomendada VARCHAR(3) NOT NULL,

duracion VARCHAR(5) NOT NULL,

sinopsis VARCHAR(4000),

director VARCHAR(100),

fechaestreno DATE,

cartel VARCHAR(4000),

valoracion FLOAT,

actores VARCHAR(4000),

PRIMARY KEY(idpelicula)

);

CREATE TABLE cinesysesionesypeliculas(

idcine INTEGER NOT NULL,

idpelicula INTEGER NOT NULL,

fecha DATE NOT NULL,

hora VARCHAR(5) NOT NULL,

PRIMARY KEY(idcine,idpelicula,hora,fecha),

FOREIGN KEY(idcine)

REFERENCES cines(idcine)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(fecha,hora)

REFERENCES sesiones(fecha,hora)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(idpelicula)

REFERENCES peliculas(idpelicula)

ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE usuarios(

iduser INTEGER NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(150) NOT NULL,

password VARCHAR(20) NOT NULL,

avatar VARCHAR(4000),

correo VARCHAR(150) NOT NULL,

PRIMARY KEY(iduser)

);

CREATE TABLE cinesyusuarios(

idcine INTEGER NOT NULL,

iduser INTEGER NOT NULL,

valoracion INTEGER,

comentario VARCHAR(4000),

PRIMARY KEY(idcine,iduser),

FOREIGN KEY(idcine)

REFERENCES cines(idcine)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(iduser)

REFERENCES usuarios(iduser)

ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE peliculasyusuarios(

idpelicula INTEGER NOT NULL,

iduser INTEGER NOT NULL,

valoracion INTEGER,

comentario VARCHAR(4000),

PRIMARY KEY(idpelicula,iduser),

FOREIGN KEY(idpelicula)

REFERENCES peliculas(idpelicula)

ON UPDATE CASCADE ON DELETE CASCADE,

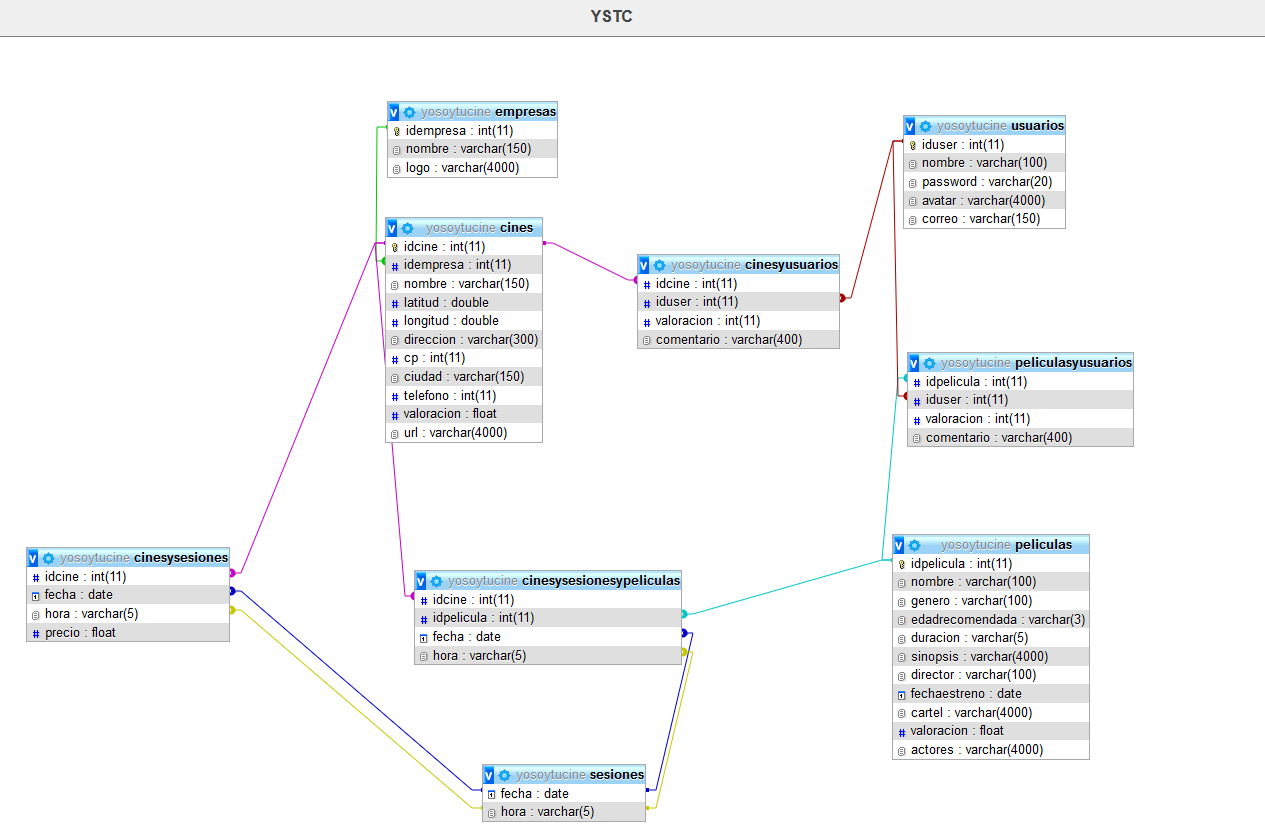
FOREIGN KEY(iduser)

REFERENCES usuarios(iduser)

ON UPDATE CASCADE ON DELETE CASCADE

);

El esquema de la BD se puede ver gráficamente en el siguiente diagrama:



**Desarrollo y despliegue de la aplicación**

La aplicación está realizada bajo el patrón de diseño Modelo-Vista-Controlador (MVC), para su codificación hemos utilizado el IDE ECLIPSE que se integra perfectamente con GIT y hemos utilizado el servidor de aplicaciones TOMCAT para desplegar la aplicación.

Para llevar a cabo el patrón MVC creamos los DAO y los BO.

A groso modo podemos decir que: el DAO es el que va a base de datos y se conecta con ella, y el BO es el encargado de insertar, borrar, actualizar etc…contiene el esquema/propiedades de la Base de datos.

Vamos a ver las definiciones de BO y DAO:

Un BO (objeto de negocio) es un tipo de entidad inteligible que es un actor dentro de la capa de negocio de un programa de ordenador basado en *n* capas.

Mientras que un programa podría implementar clases, las cuales pueden ser objetos controlando o ejecutando comportamientos, principalmente se distinguen en que no realizan nada por sí mismos, sino que albergan un conjunto de atributos y asociaciones con otros, tejiendo un mapa de jugadores que representan las relaciones de negocio.

Un DAO (objeto de acceso a datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. El término se aplica frecuentemente al Patrón de diseño Object.

Ventajas:

Los Objetos de Acceso a Datos son un Patrón de los subordinados de Diseño Core J2EE y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

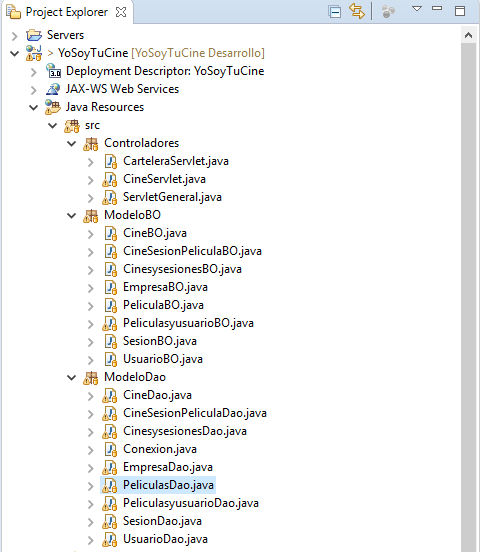
Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente (API de Persistencia Java), la cual podría ser JDBC, JDO, Enterprise JavaBeans, TopLink, EclipseLink, Hibernate, iBATIS, o cualquier otra tecnología de persistencia. Usar Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

Desventajas:

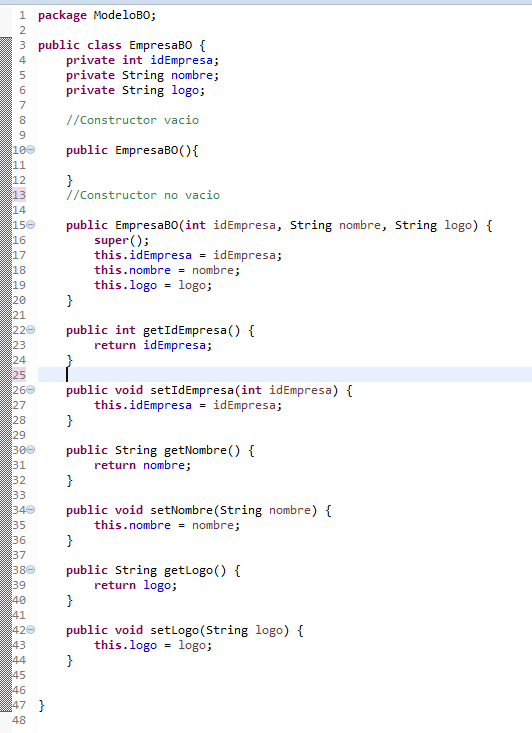
La flexibilidad tiene un precio. Cuando se añaden DAOs a una aplicación, la complejidad adicional de usar otra capa de persistencia incrementa la cantidad de código ejecutado durante tiempo de ejecución. La configuración de las capas de persistencia requiere en la mayoría de los casos mucho trabajo.

Una vez vistas estas explicaciones vamos a ver algunas capturas y código interesante de nuestra aplicación.

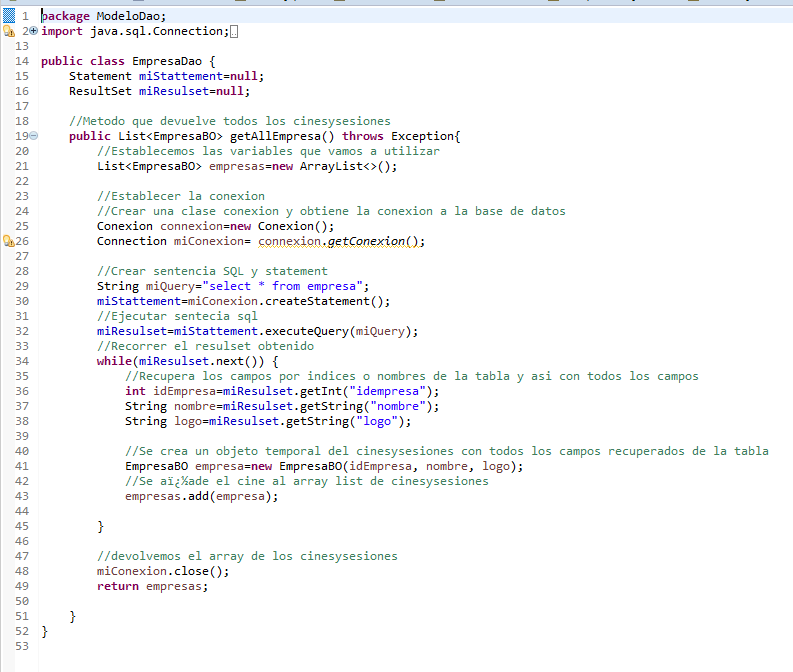
**Estructura del proyecto web JAVA con MVC**



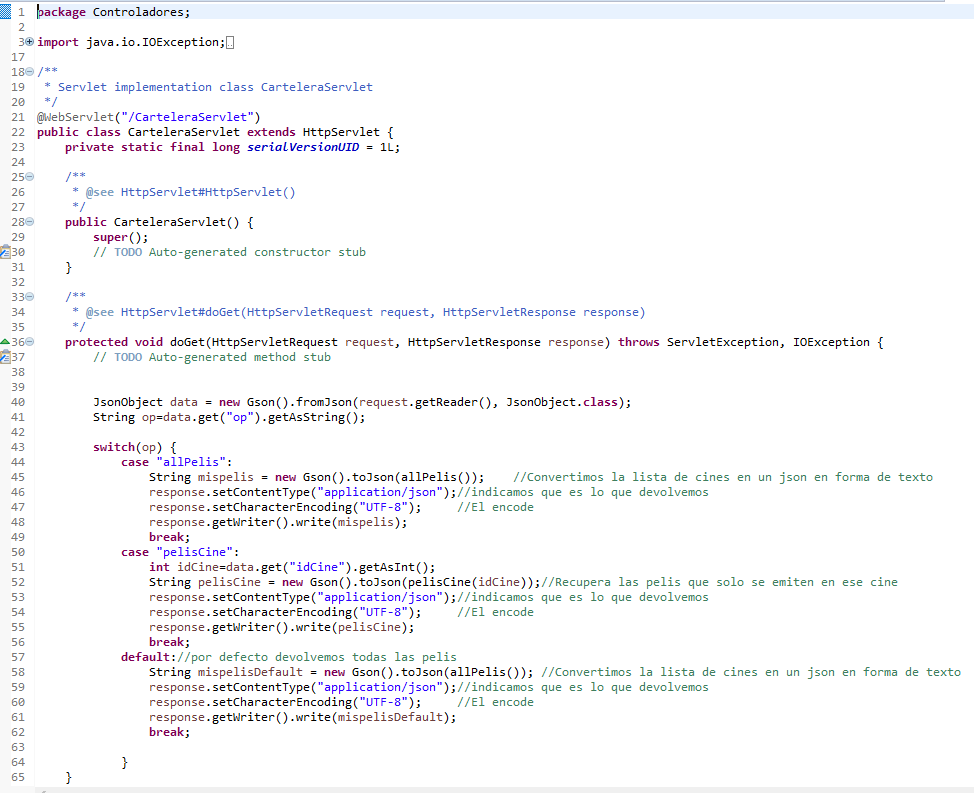
**BO**



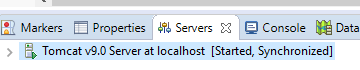
**DAO**



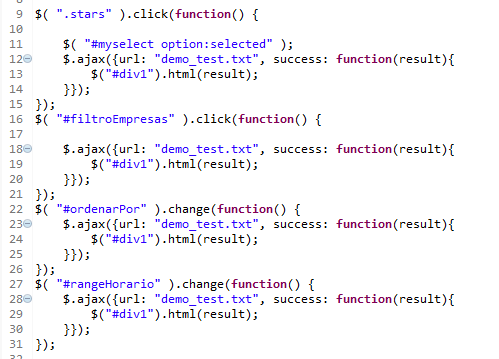
**Servlets**



**Servidor TOMCAT**

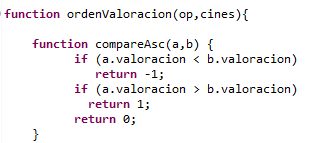


**Llamadas Ajax**

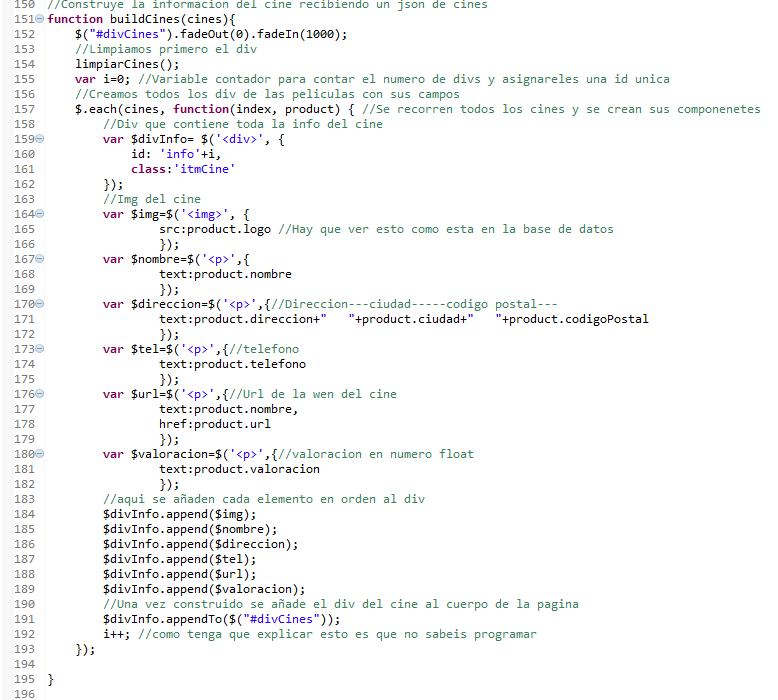


**Métodos Java y JS en los filtros de las web**

****



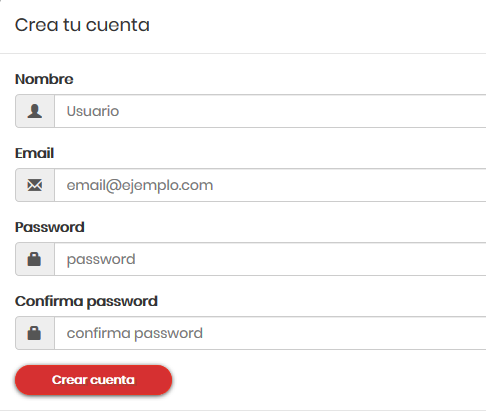
**Generación de HTML mediante JQuery**

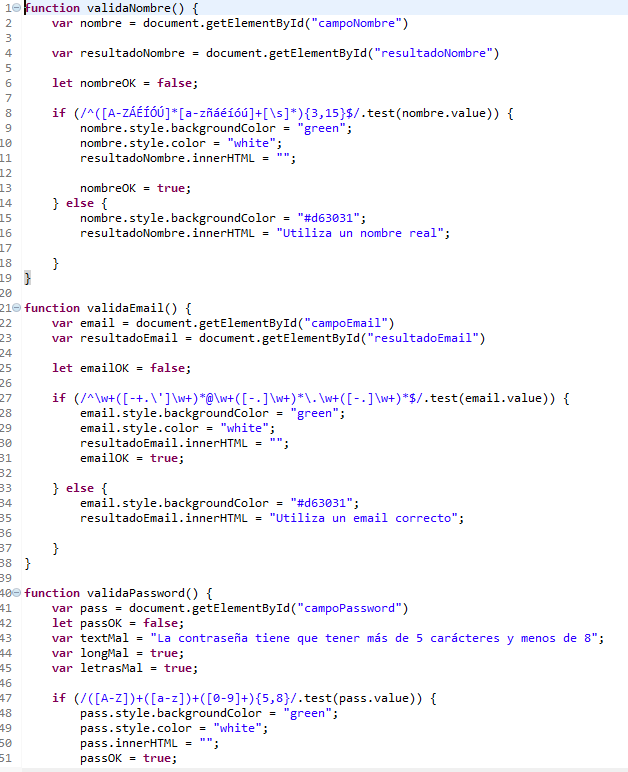


**Captura de la web funcionando**

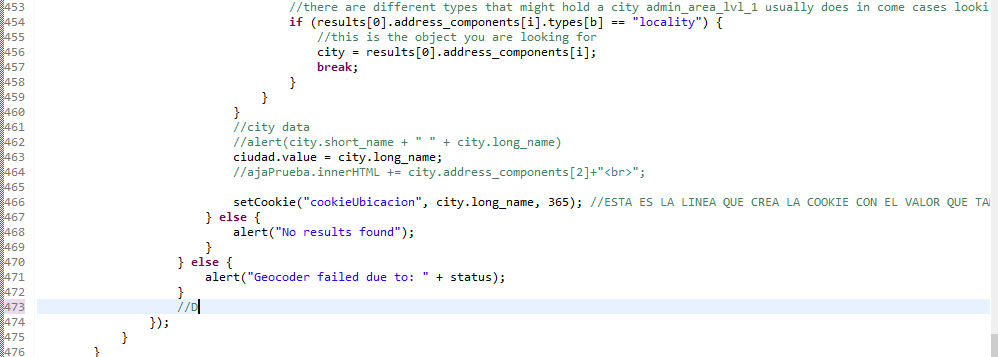


**Validaciones formulario registro**

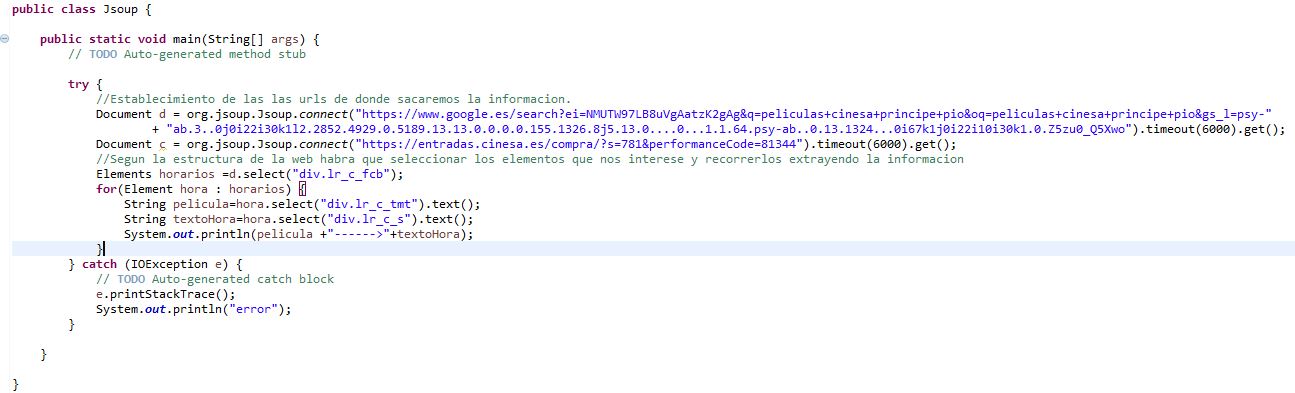


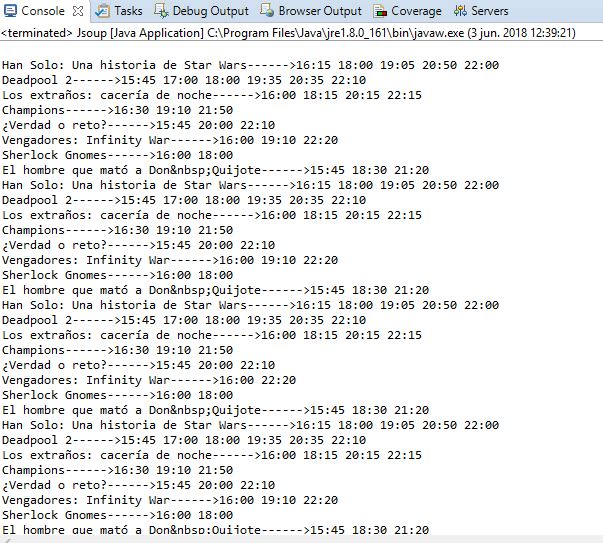


**Geolocalización y cookie**



**Scrapping**





**Bibliografía**

<https://es.wikipedia.org/wiki/Objeto_de_transferencia_de_datos>

<https://es.wikipedia.org/wiki/Objeto_de_acceso_a_datos>

<https://es.wikipedia.org/wiki/Web_scraping>

https://developers.google.com/maps/documentation/geolocation/intro?hl=es-419

<http://codejavu.blogspot.com/2013/06/ejemplo-modelo-vista-controlador.html>

<http://blog.hostdime.com.co/agregar-y-eliminar-elementos-de-forma-dinamica-con-jquery/>

https://es.stackoverflow.com/questions/40754/crear-elementos-del-dom-con-jquery