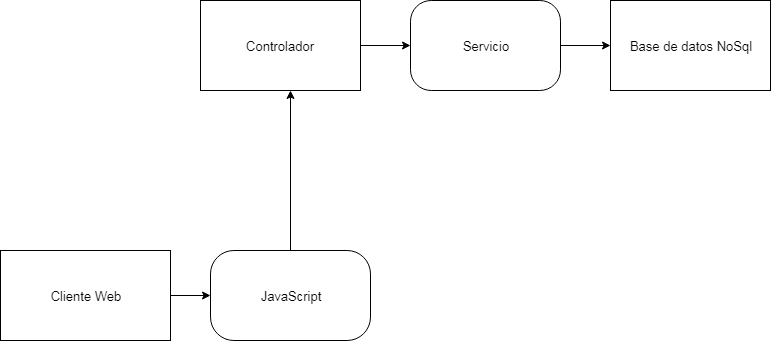
**KANBAN SIMULATOR**

Arquitectura

La aplicación se desarrolla en un proyecto Maven donde la estructura es la siguiente:

* La parte front-end donde usamos JavaScript, Jquery, Chart.
* La parte back-end donde usamos Java-Spring
* Base de datos NoSql RestDB basada en Mongo.



Servidor

La parte del servidor se estructura de esta manera:

* Controlador:
  + HomeController.
* Servicio:
  + DistributionService : Genera los valores Poisson, Gaussian, Weight para la distribución de tareas.
  + RestdbService: Hace peticiones Get ( find ), Post ( save ), Put ( update ).
* Listener:
  + SessionCounter: Crea, destruye y genera las visitas de usuarios por navegador.

La base de datos RestDB tiene la siguiente estructura:

* NameBlueprint : Nombre que recibe la plantilla.
* listUsers : Objeto que contienen todos los datos de los usuarios que trabajan.
* listPhases: Objeto que contiene la configuración de las fases.

Front - JavaScript

## Kanban.js

Este archivo es quien controla la parte principal de logística del tablero Kanban, en él se encuentran las siguientes funciones:

* play(): Poner en funcionamiento el Kanban y de hacer los cálculos y comprobaciones de tareas, asignaciones de usuario, distribución y tiempos estimados.
* deshabilitarMenus(bool disable): Controlar el acceso a los diversos elementos interactivos de la el tablero este en funcionamiento.
* shortPhases(): Permitir el desplazamiento las fases para poder cambiar el orden de estas.
* getGaussian(): Realiza una petición al servidor para que este le devuelva un nuevo valor Gaussiano para controlar la entrada de tareas en distribución normal.
* getPoisson(): Realiza una petición al servidor para que este le devuelva un nuevo valor Poisson para controlar la entrada de tareas en distribución posson.
* getWeight(): Realiza una petición al servidor para que este le devuelva dos valores, uno aleatorio con un rango configurable y un segundo valor que determina el “peso” que tendrá la tarea en distribución con peso.
* getDistribution(): Reasignar los últimos valores introducidos en la pantalla de configuración de la distribución.
* PrintTasks(): Añadir los elementos HTML que corresponden a los objetos “Tarea” generador para que sean visibles en el tablero.
* stopWatch(): Muestra el tiempo transcurrido desde el inicio de la ejecución del tablero Kanban.
* timer(): Mostrar el tiempo restante especificado del tablero Kanban.
* speedKanban(String value): Controlar la velocidad a la que se ejecuta el tablero Kanban.
* loadCharts(): Crear una vista por pestañas de las gráficas.

## backlogconf.js

Se encarga de controlar la ventana de configuración de la distribución de la entrada de tareas al tablero.

## Resultados.js

Sirve como distribuidor y da paso a las funciones que el usuario clica:

* mostrarResultados : Muestra los resultados en el navegador.
* generarResultados : Genera los resultados obtenidos en el tablero.
* mostrarGraficas : Muestra las gráficas en el navegador
* generarGraficos : Genera los gráficos.
* mostrarKanban : Muestra el tablero de kanban.
* mostrarFinalTarea : muestra cyleTime y leadTime al terminar la tarea.

## ResultadosService.js

Actúa como servicio para la generación de resultados, dando paso a las funciones principales:

* tableTask : Muestra los resultados de las tareas.
* behindTask: Muestra el apartado “ ver más datos “ de la tabla de tareas.
* tablePhase: Muestra los resultados de las fases.
* tableUser: Muestra Muestra los resultados de los usuarios del kanban.
* behindUser: Muestra el apartado “ ver más datos ” de la tabla de usuarios de kanban.

También contiene funciones secundarias para la generación de resultados como:

* nuevoObjetoResultados : Crea un objeto resultados.
* rellenarResultados: Guarda los resultados obtenidos en el objeto.
* buscarMasTrabajador : Devuelve al trabajador que más o menos ha trabajado en segundos pasando por parámetro ‘max’ o ‘min’.

calcularMediaCycleAndLead : Calcula las medias de cycle y lead time.

* ultimas20TareasCTyLT: Calculas las medias de cycle y lead time de las ultimas X tareas.
* updateGraficPhase : Modifica el gráfico de fases.
* calculoTiemposTotalesFase : Calcula el tiempo total de cada fase.
* saveTimeStates : Calcula el tiempo de cada estado de la tarea.
* sumaFasesTiempo: Suma el tiempo de las fases.
* saveNewTimePhase : Suma el tiempo de los estados por tarea.
* calcularMediaPorTarea : Calcula la media de estados de la tarea.
* mediaFasestotal :Calcula la media de las fases.
* findMaxAndMin : Busca los segundos de los usuarios que han trabajado más o menos.
* maxAndMinUsers : Busca los nombres de los usuarios que han trabajado más o menos.
* getRandomColor : Genera un color aleatorio.
* saveBlueprint : Petición Ajax para el guardado de resultados en la base de datos.
* getBlueprints : Petición Ajax para recuperar los resultados en la base de datos.
* getNameBluePrint: Guarda las plantillas en un array.
* updateBlueprints: Petición Ajax para modificar los resultados en la base de datos.

## Modals.js

Es la página encargada de toda la funcionalidad en los “modals”, ventanas emergentes, etc.

Dentro del cual podemos reconocer:

* InitializeAddPhases(), InitializeModPhases(), InitializeAddUsers(),initializeModUsers():   
  Sirven para inicializar los formularios de creación y modificación de fases y usuarios.
* SaveAddUser(), SaveModUser(), SaveAddPhase(), SaveModPhase() :

Sirven para guardar sesión, tanto los nuevos usuarios y fases, como las modificaciones de los mismos.

* RmvModUser();

Elimina por completo a un usuario de la sesión

* FormUserValido(): Encargado de comprobar que los datos introducidos en el formulario de usuarios sean correctos.
* addInput() , InsertInput():

Añaden dentro del formulario los “sliders” para modificar el rendimiento de recursos por fase.

* addTareas():

Añade una nueva tarea dentro del kanban

* showTaskInfo():

Muestra la información detallada de cada tarea cuando se clica sobre ellas

* chrono():

Inicializa el cronometro y lo formatea para que tenga el formato “mm:ss”.

* loadSelectedBlueprintnames():

Muestra la lista con todas las plantillas disponibles

* emptyDropDownList():

Vacia la lista de las plantillas actuales

* ValidateNameBlueprints():

Comprueba si no existe en base de datos una plantilla con ese mismo nombre, si es así te la guarda, pero si no, te pide confirmación para sobrescribir