## horizontal line



Experimento Numero 1

15.02.2017

**─**

Juan Pablo Otalora cod: 201425525

Ana María Espinosa cod: 201425031

Fabio Andrés López cod: 201423782

Grupo numero 2

Universidad de los Andes

Bogotá, Colombia

# Vision

Este documento se basa en la idea de discutir las decisiones arquitectónicas que tomamos en el desarrollo de nuestro primer experimento. Para ello documentamos las decisiones y el porqué de estas mostrando lo que nos dieron pruebas de carga sobre la primera iteración.

# Objetivos

1. Desarrollar una arquitectura válida para la solución del problema dado.
2. Escribir el código necesario para cumplir los requerimientos dados.

# Especificaciones

Para el inicio de este documento cabe resaltar que como primera decisión que tomamos fue el uso de Java Play como el framework principal de nuestra aplicación. La elección de Java Play fue basada en la idea de encontrar soluciones viables para nuestro proyecto. Java Play permite el uso de un framework asincrónico y no bloqueante, el cual nos ayuda en entregar y pasar información al cliente de una manera más eficiente. Funciona además con un modelo de Cliente/Servidor que nos ayuda con el acceso a data compartida y centralizada. Por último nos ayuda con la escalabilidad ya que escala horizontalmente que facilita la posibilidad de incrementar la capacidad rápidamente.

## Conclusion acerca del Framework

Java Play es un framework moderno que usa múltiples modelos de arquitectura de software que se amoldan de una gran manera para el problema propuesto en clase. Además se aprende un nuevo lenguaje y se gana una mayor experiencia.

# Problematica:

Para comenzar debemos contextualizar el caso propuesto para el desarrollo. Nos encontramos desarrollando una aplicación web como una solución para un hospital cardiaco. Debemos poder ayudar a los pacientes avisando a servicios de emergencia cuando el ritmo cardiaco tenga cambios abruptos, además queremos que la aplicación pueda manejar registros de estos casos y de los casos normales para que los médicos puedan dar consejos correctos y cambiar el marcapasos en caso de necesitarlo. Por todo lo anterior nos damos cuenta que estamos jugando con la vida de las personas lo cual requiere medidas especiales.

## Hipotesis

En este caso nuestro web app debe priorizar la rapidez en el tiempo de respuesta y el desempeño de la aplicación por encima de todo. La aplicación debe ser capaz de alertar a los médicos y servicios de emergencia con rapidez y exactitud el caso en el cual algún paciente se encuentre en peligro.

# Descripción del experimento

En el experimento queremos construir los cimientos de nuestra aplicación para poder observar tiempos de respuesta sobre nuestros requerimientos. La configuración en JMeter se realiza con 3000 threads y un ramp-up de 1 segundo.

# Requisitos para el desarrollo del experimento

## Artefactos a construir

Para la construcción de la aplicación se usa la estructura básica requerida para el uso del framework Play. Por parte del framework se estarán usando los Streams de Akka para manejar transacciones asíncronas, junto a librerías en Scala que posteriormente manejarán diferentes elementos de la base de datos y de los templates. Así mismo es clave el uso de SBT, herramienta que maneja las dependencias y permitirá generar el build de la aplicación.

Es necesario desarrollar, bajo la misma estructura estándar, diferentes módulos haciendo uso de las APIs que se están implementando. Para contestar los servicios HTTP está el archivo de routes (ubicado en conf/routes dentro de la carpeta madre del proyecto), quien se encarga de asociar el request generado por el cliente con un método desarrollado en el controller del recurso. Estos controllers serían uno por cada módulo a usar dentro de la aplicación (inicialmente todas las entidades). El controller se comunica con la parte de lógica, donde se almacenan los archivos .java que representan a las diferentes entidades lógicas involucradas en el modelo. En esta etapa inicial se involucran todos estos módulos para generar los CRUD y los requerimientos necesarios.

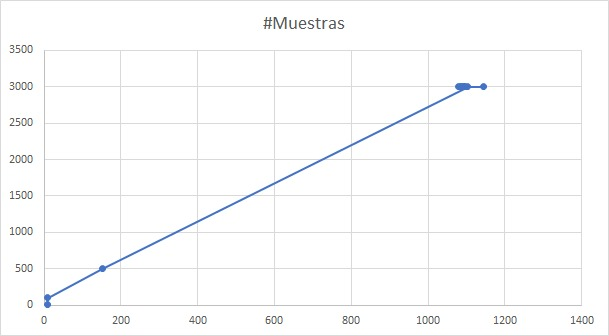
1. Recursos de la experimentación

Para la ejecución de la aplicación se utiliza el IDE Intellij 2016.3.4, que despliega los servicios web de la aplicación. Previo a la experimentación se implementan los servicios HTTP requeridos desde el complemento de google chrome POSTMAN para probar el funcionamiento correcto de la aplicación. El despliegue se realiza en un equipo con un procesador Intel de 2.60GHz, con una memoria RAM de 8GB en un sistema operativo de 64 bits. Estas pruebas hacen uso de la herramienta Apache JMeter (3.1.r1770033).

# Planeación y resultados esperados

Para una configuración de 3000 peticiones con un ramp-up de un segundo se espera un comportamiento que no exceda los dos segundos. En cada etapa del experimento se irá aumentando el número de peticiones gradualmente hasta llegar al número esperado. Todos los datos se anotarán y se verá el comportamiento de la aplicación ante el número máximo de peticiones esperado.

# Resultados obtenidos

Tras hacer múltiples versiones de la prueba de carga se obtienen los resultados observados en la gráfica:

Como se puede ver, se organiza la duración de las transacciones (en milisegundos) y se observa cuántos threads fueron implementados para ese resultado. Para valores bajos del número de muestras (de 500 o menos) se tiene un tiempo inferior al de 200 milisegundos, un resultado ideal. Sin embargo, se nota un crecimiento lineal al aumentar el número de threads. Al llegar al número deseado (3000) se realizan varias pruebas, las cuales muestran un comportamiento constante del tiempo de respuesta, que siempre ronda los 1100 milisegundos. Los resultados son los esperados.

# Conclusiones

* Los elementos arquitectónicos de actores permiten un funcionamiento más eficiente de los proyectos con alto número de peticiones. Al realizar un trabajo asincrónico sobre las peticiones que llegan, se logra aprovechar un mayor porcentaje de los recursos computacionales con los que se cuentan. Es evidente que cuando se tiene una configuración de prueba dada, el modelo no bloqueante funciona como es debido y el porcentaje de transacciones se hace más cercano a cero.
* Debemos mejorar la escalabilidad, el sistema debe ser capaz de responder múltiples peticiones de los clientes en un tiempo rápido (1 segundo) para la resolución de peticiones y registros por parte de los clientes.