



SemeruSocial: FRIENDMATCH

Informe de Cierre de Proyecto y lecciones aprendidas

Versión: 003

Fecha: 20/12/2019

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

HOJA DE CONTROL

Organismo	Semeru Social		
Proyecto	Friendmatch		
Entregable	Informe de Cierre de Proyecto y lecciones aprendidas		
Versión/Edición	003	Fecha Versión	20/12/2019
Nº Total de Páginas	35	Fecha Aprobación	20/12/2019

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
001	Versión inicial	Mirian Isabel Martínez Álvarez	19/12/2019
002	Tests De Rendimiento	Ishwara Coello Escobar	20/12/2019
003	Deploy y módulos	Mirian Isabel Martínez Álvarez	20/12/2019

ÍNDICE

1 INTRODUCCIÓN.....	5
1.1 Objeto.....	5
1.2 Alcance del proyecto.....	5
2 GESTIÓN: HERRAMIENTAS.....	6
2.1 Eclipse.....	6
2.2 Git.....	6
2.3 Redmine.....	6
2.4 Maven.....	6
2.5 Jenkins.....	6
2.6 SonarCube.....	7
2.7 Profilers.....	7
2.8 JMeter.....	7
2.9 Errores destacables.....	8
3 CONSIDERACIONES FASE A FASE.....	9
3.1 ITERACIÓN 1.....	9
3.1.1 Alcance.....	9
3.1.2 Organización del proyecto y documentación.....	9
3.1.3 Consideraciones:.....	9
3.1.4 Desviación del proyecto.....	9
3.2 ITERACIÓN 2.....	10
3.2.1 Alcance.....	10
3.2.2 Organización del proyecto y documentación.....	10
3.2.3 Consideraciones:.....	14
3.2.4 Desviación del proyecto.....	14
3.3 ITERACIÓN 3.....	15
3.3.1 Alcance.....	15
3.3.2 Organización del proyecto y documentación.....	15
3.3.3 Consideraciones:.....	18
3.3.4 Desviación del proyecto.....	19
3.4 ITERACIÓN 4.....	20
3.4.1 Alcance.....	20

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

3.4.2 Organización del proyecto y documentación.....	20
3.4.3 Consideraciones (e informe):.....	20
3.4.4 Desviación del proyecto.....	30
4 ASPECTOS RELEVANTES.....	31

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------	-------------------------------------------

1 INTRODUCCIÓN

1.1 Objeto

Este documento tiene como propósito proporcionar un punto fijo en el que el proyecto ha alcanzado los objetivos y producto solicitado.

1.2 Alcance del proyecto

El proyecto abarca las características y funcionalidades que caracterizan el producto, pedidas en cada iteración (ver. Apartado de fase a fase). Así como el uso adecuado de las herramientas pedidas: IDE Eclipse, control de versiones Git, sistema de tareas Redmine, empaquetado Maven, Integración continua Jenkins, inspección continua con SonarQube, monitorización con profilers, pruebas de rendimiento de Jmeter.

La aplicación esta deployeado en deploy.fic.udc.es/semerusocial/

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------	-------------------------------------------

2 GESTIÓN: HERRAMIENTAS

2.1 Eclipse

Se ha usado Eclipse como IDE y centralización de las demás herramientas; así como la configuración de save-actions comunes a todos los miembros.

2.2 Git

Se ha empleado como estrategia “feature branching”, manteniendo cada funcionalidad y sus cambios en una rama. Una vez la rama está completa (está terminada la funcionalidad, testeada y pasa sonar) se abre una Merge Request la cual debe ser revisada y mergeada por el miembro más adecuado para la funcionalidad.

Cada commit debe ser completo, con descripción adecuada e idealmente con el comentario pertinente para redmine (tarea y tiempo a imputar).

2.3 Redmine

Se ha empleado Redmine para la gestión de tareas (features, issues y support), estimándoles un tiempo inicial y asignándose a distintos miembros. Se ha usado target versions para identificar la iteración a la que pertenecen las tareas y la Wiki como documentación inicial de la iteración para el backend.

2.4 Maven

Uso de maven para el empaquetado y gestión de dependencias y plugins, incluyendo aquellas necesarias e incluyendo dependencias transitivas para forzarles una versión concreta (por ejemplo, commons-logging).

También se ha usado maven para la definición de perfiles para el uso de las bases de datos y para los servers embebidos.

Así mismo se han eliminado aquellas dependencias que ya eran recibidas por otras.

2.5 Jenkins

Para la integración continua se emplea Jenkins usando un hook de git para ello proporcionado por el profesorado. Se ha configurado en éste los permisos (todos para todos los miembros del equipo).

Se ha configurado la conexión al repositorio de git para que se builden todas las ramas

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

cada vez que se envía un commit debido a la estrategia seguida de branching.

Se ha añadido un script pre-step para configurar las ramas de sonar para compararlas con develop o master según se necesite. Para la build se usan como perfiles h2 y jetty asignándole a este último el puerto 13131 por defecto. Y como post-step se ha añadido la ejecución de sonar pasándole las propiedades necesarias.

Después de buildear se almacena el war y se le pone una tag en git con el id de la build.

Con multimódulo simplemente hay que añadir la propiedad de sonar para módulos. A la hora de buildear se generan tantos wars como módulos con el nombre por defecto o asignado en el pom en nuestro caso. Para deployar se le especifica que wars tomar (o que tome todos) y por defecto lo publica en el contexto con el nombre del war (otra opción sería un archivo de configuración del contexto).

2.6 SonarCube

Se ha dejado la configuración por defecto de Sonar aunque inicialmente se excluyó la parte de webapp de la coverage.

Se ha usado para ver bugs, issues y su resolución, así como la coverage y el code duplication todo como apoyo para la decisión de la merge request.

2.7 Profilers

Se usó JMC para inspeccionar y analizar el estado de la aplicación durante la ejecución de esta.

Este análisis incluye sobre todo la búsqueda de *memory leaks*.

2.8 JMeter

Herramienta utilizada para probar la aplicación ante distintas cargas y así poder detectar los límites efectivos que puede soportar la aplicación así como probar la eficiencia de distintas configuraciones.

2.9 Errores destacables

Se excluyó inicialmente el webapp del coverage scanner de Sonar.

Las ramas de features tienen varios commits adicionales debido a la subdivisión de la feature.

	<p>Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p>SemeruSocial</p>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	----------------------------

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------	-------------------------------------------

3 CONSIDERACIONES FASE A FASE

3.1 ITERACIÓN 1

3.1.1 Alcance

Fecha de entrega: 04/10/2019

Descripción de los objetivos:

“You should be able to upload pictures to your personal page where the pictures will be displayed in a carousel. Under the picture you will have the opportunity to add a personal description and metadata (age, sex, city). Pictures will only be visible to the owner when logged. Pictures can be removed or added and description can be edited at any time by the owner.”

Estado final: Aceptado.

3.1.2 Organización del proyecto y documentación

No aplica.

3.1.3 Consideraciones:

FUNCIONALES: La metadata había sido asignada a la fotografía en vez de al usuario.

3.1.4 Desviación del proyecto

Aún no se realizaba estimación de las tareas.

Entregado en tiempo.

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------	-------------------------------------------

3.2 ITERACIÓN 2

3.2.1 Alcance

Fecha de entrega: 25/10/2019

Descripción de los objetivos:

“You can specify friend search criteria: sex (male, female, either), cities, age (range). You should have a “Find a friend” tab where users matching your search criteria will be displayed one-at-a-time. You will only be able to see a new friend suggestion after accepting or discarding the current suggestion. Friendsuggestion will show the same information as the user profile currently being suggested. There will be two buttons one for accepting the suggestion and other for rejecting it. Rejected suggestions will never be displayed to you again. There will be a new tab “We are friends” where a list of your friends will be displayed. Two users to be friends means that both users accepted the suggestion of the other user.”

Estado final: Aceptado.

3.2.2 Organización del proyecto y documentación

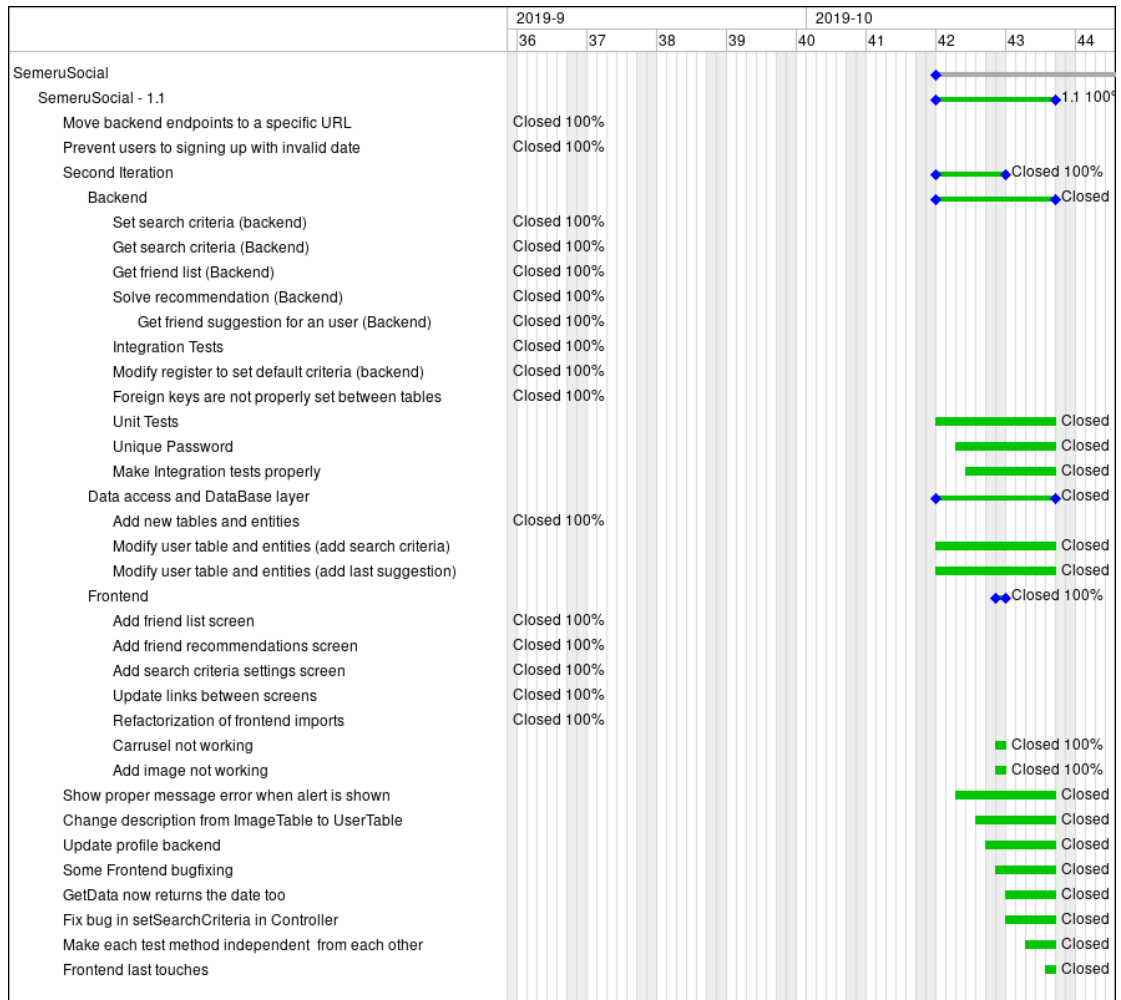
Tag redmine: [SemeruSocial – 1.1](#)

División en tareas pero no con tiempo estimado.



Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas

SemeruSocial



Documentación de diseño inicial añadida en la wiki de redmine:

<https://redmine.fic.udc.es/projects/semerusocial/wiki>

PERSISTENCIA

REQUEST (RequestTable)

subject (Remitente)

object (El que recibe la petición)

date

MATCH (MatchTable)

--Ordenada por id User1<User2

user1

user2

date

REJECTED (RejectedTable)

subject (el que rechaza)

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

object (el rechazado)

date

USER (UserTable)

--NOTA: Poner una por defecto a los usuarios ya creados

criteriaSex --Por defecto 'ANY'

criteriaMinAge --Por defecto 18

criteriaMaxAge --Por defecto 99

CITIESCRITERIA (CitiesCriteria)

--NOTA: Si no hay ninguno para el usuario es como un any

userId (P)

cityName (P)

DTOS

FriendDto

int age

String sex

String city

String name

String description

SearchCriteriaDto

SexCriteriaEnum sex -- "female", "male" or "other"

int minAge

int maxAge

List <String> cities

(Title: Edit this section)

Edit this section

SERVICES

FriendService

suggestFriend (Long userId) : Optional<FriendDto>

--NOTA: Si no encuentra ninguno sería null

--NOTA: En cuanto a criterio de genero si es otro vale cualquier cosa estilo "patata", "hola", etc, cualquier cosa que no sea: "female" "male"

-INFE

acceptRecommendation (Long subject, Long object): boolean

--(Devuelve 1 si es match, 0 si no)

-INFE --Comprobar si existe el usuario antes de llamar a la funcion privada

InvalidRecommendationException

AlreadyRejectedException

rejectRecommendation (Long subject, Long object): boolean

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------	-------------------------------------------

-INFE --Comprobar si existe el usuario antes de llamar a la funcion privada
InvalidRecommendationException
AlreadyRejectedException

CriteriaService

SetDefaultCriteria (Long userId) : void

-INFE

GetCriteria (Long userId): UserCriteriaDto

-INFE

SetCriteria (UserCriteriaDto criteria, Long userId): void

-INFE

-ValidationException

--si maxAge<minAge

MatchService

GetUserFriends (Long userId, int page, int size): Block<FriendDto>

INFE

MaximumSizeExceededException --size maximo 50

CONTROLLERS

SuggestionsController

GET /backend/suggestion

public suggestFriend: FriendDto

@RequestAttribute Long userId

INFE

POST /backend/suggestion

public solveFriendSuggestion: boolean

@RequestBody accept

@RequestAttribute Long userId

INFE

h3. UserController

GET /backend/users/criteria

public getUserCriteria: UserCriteriaDto

@RequestAttribute Long userId

INFE

	<p align="center">Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p align="center">SemeruSocial</p>
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------	-------------------------------------------

POST /backend/users/criteria

public setUserCriteria: UserCriteriaDto

@RequestAttribute Long userId

@RequestBody UserCriteriaDto

INFE

ValidationException

h3. MatchController

GET /backend/users/friends?page=

public getFriends: BlockDto<FriendDto>

@RequestAttribute Long userId

@RequestParam page

INFE

MaximumSizeExceededException

NOTA: size=20

3.2.3 Consideraciones:

HERRAMIENTAS: No se realizó la dependencia entre tareas de muchas de ellas, aunque no todas.

FUNCIONALES: Edad mínima debía ser 18 años.

3.2.4 Desviación del proyecto

Stimated time [95.25 hours](#)

Spent time [102.93 hours](#)

Entregado en tiempo.

3.3 ITERACIÓN 3

3.3.1 Alcance

Fecha de entrega: 22/11/2019

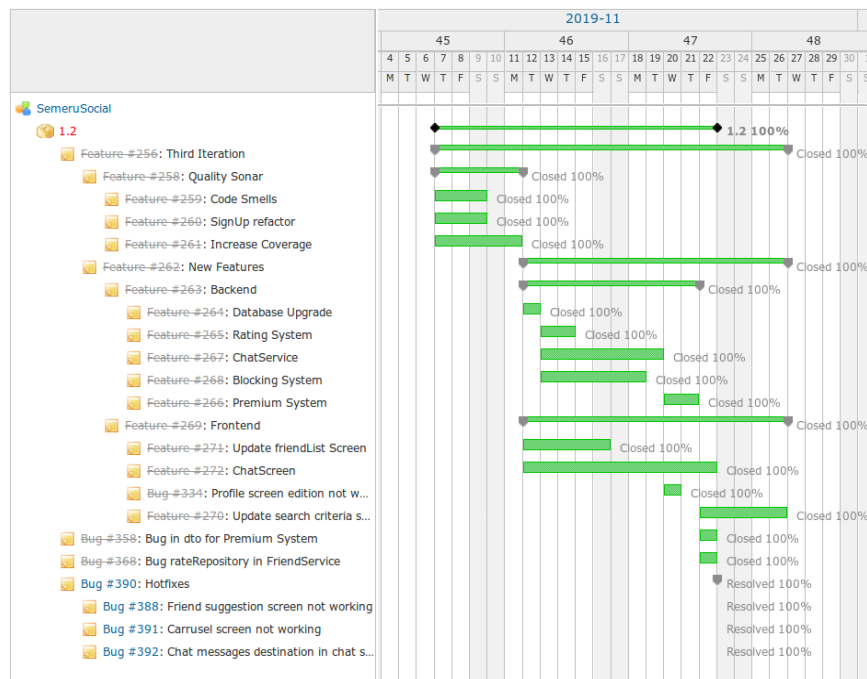
Descripción de los objetivos:

“ Now, you can only see friend suggestions of users with which you match with their search criteria. You can start chat and talk with the users you are friends with. Chat content is stored in the server. You can block a friend, meaning you are not friends any more and that user will not be suggested to you in the future. You can rate your friends in a scale from 1 to 5. Average rating of a user is shown in their profile page. Users will have the option to specify in their search criteria a minimum rating. Users will only be able to specify this if they have been rated by someone. Users can specify a minimum rating only one point up over their current rating. Users can pay to be premium users. Premium users can use any filter they want in any way. Premium users will be randomly shown as friend suggestions to any user matching their criteria even when they do not match the search criteria of those users.”

Estado final: Aceptado.

3.3.2 Organización del proyecto y documentación

Tag redmine: [SemeruSocial - 1.2](#)



DOCUMENTACIÓN: Documentación de diseño inicial de backend: [Wiki redmine 3rd iteration](#)

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

TABLAS

UserTable

-rating (double) --CONSTRAINT CHECK (1<=r<=5)

--NOTA: Nullable

-ratingVotes (long) --CONSTRAINT check >=0

--Cantidad de votos recibidos

-premium bool

-minRateCriteria (int)

MessageTable

-messageId

-user1

-user2

-emisorId (FK)

-messageContent --VARCHAR(1000)

-date (FECHA + HH:MM:ss.milis)

RateTable

-rateId

-subjectId (FK) (El que vota)

-objectId (FK) (El votado)

-points (Int entre [1 y 5])

BlockTable

-blockId

-subjectId

-objectId

SERVICES

USERSERVICE

-void rateUser (long subjectId, long objectId, int rating)

//INFE

//VE (ValidationException) (si rating<1 o rating>5)

//NotYourFriendException

--NOTA: Actualizar el minRateCriteria según el rate medio

-void updatePremium (long userId, boolean isPremium)

//INFE

FRIENDSERVICE

-void blockUser(long userId, long friendId)

//INFE

//AlreadyBlockedException

//NotYourFriendException

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

NOTA: Dejan de ser amigos

-void sendMessage(long userId, long friendId, String content)

//INFE

//NotYourFriendException

//ValidationException (tam messageContent)

-Block getConversation (long userId, long friendId, int page, int size)

//INFE

//NotYourFriendException

MessagesDetailsDto

-contenido

-emisor

-fecha

CONTROLLER

/backend/users/rating POST

ENTRADA:

int puntuacion (min 1, max 5)

userId

friendId

SALIDA: NA

CODE: NO_CONTENT (204)

/backend/premium PUT

ENTRADA:

userId

makePremium: boolean

SALIDA:

NA

CODE: NO_CONTENT (204)

/backend/friends/block POST

ENTRADA:

userId

friendId

SALIDA: NA

CODE: NO_CONTENT

/backend/friends/conversation GET

ENTRADA:

userId

friendId

page

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

sizeSALIDA: Block

CODE: OK

/backend/friends/conversation POST

ENTRADA:

userId

friendId

content

SALIDA: NA

CODE: NO_CONTENT

CAMBIOS

-Cambiar lógica de buscar suggestion friend

(Ver sugerencia si también cumples su criteria

Sugerir también premiums que no están bloqueados/rechazados/aceptados)

NOTA: si tu rate es null -> apareces en todos

-Añadir al coger la info del perfil del usuario y de un amigo-> su rate

-Añadir al ver tu criteria -> premium

-Criteria -> tiene ahora min Rate

-> InvalidMinRateCriteriaException

-> NotRatedException

-Añadir al setear y obtener criteria -> minRate (Nota: no puede ser mayor que tu propio rate+1 y si has sido valorado a no ser que seas premium)

MIRAR

-Salida de /backend/friends/conversation

-Comunicación chat

3.3.3 Consideraciones:

HERRAMIENTAS: Se ignora la parte de webapp de la coverage de sonar

FUNCIONALES: Falta ajustar feature para que al sugerir amigos se compruebe la criteria en ambos lados.

Encontrado bug de que te permitía registrarte con más de 3 años.

3.3.4 Desviación del proyecto

Stimated time [93.00 hours](#)

	<p>Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p>SemeruSocial</p>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	----------------------------

Spent time [85.66 hours](#)

Entregado en tiempo.

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

3.4 ITERACIÓN 4

3.4.1 Alcance

Fecha de entrega: 20/12/2019

Script Inicial de Base De Datos

Se ha creado 10 usuarios: User100, User101, User102, User103, User104, User105, User106, User107, User108, y User109.

Todos los usuarios tienen la contraseña 1234.

El usuario 100 tiene como amigos al User101, User102, User103, User104 y User105.

El usuario 100 tiene como usuarios bloqueados al User106 y User107.

El usuario 100 ha realizado una petición de amistad al User102 y User109.

Descripción de los objetivos:

“Your company manager decided to use the data of the users for feeding an advertisement system. For that reason, you need to provide a REST API that will allow searching for any of the user metadata and by keyword. The rest service will also expose an endpoint providing a ranking with the best-rated users in a city. Once that your REST service is ready, you should use it in the web-app for dealing with the user suggestions, so the REST API should also receive a #Iter parameter for representing the target user. Moreover, the landing page of the app will now show an anonymized carousel of photographs of the best-rated users in the city from where the web browser is open (using the REST API). Non-premium users will receive now suggestions to broaden their criteria: “If you increase your age criteria to 40, you will receive 20 more suggestions”. Your manager wants to know for the different load situations the average response time of your service for different types of queries. The application should be studied for any memory or processing leak. You should be able to deploy your application as the final step of your dev-op pipeline. With the information and data stored during the project development in the different tools fill a final report including the findings of the performance tests. Please take a look to your acceptance tests coverage.”

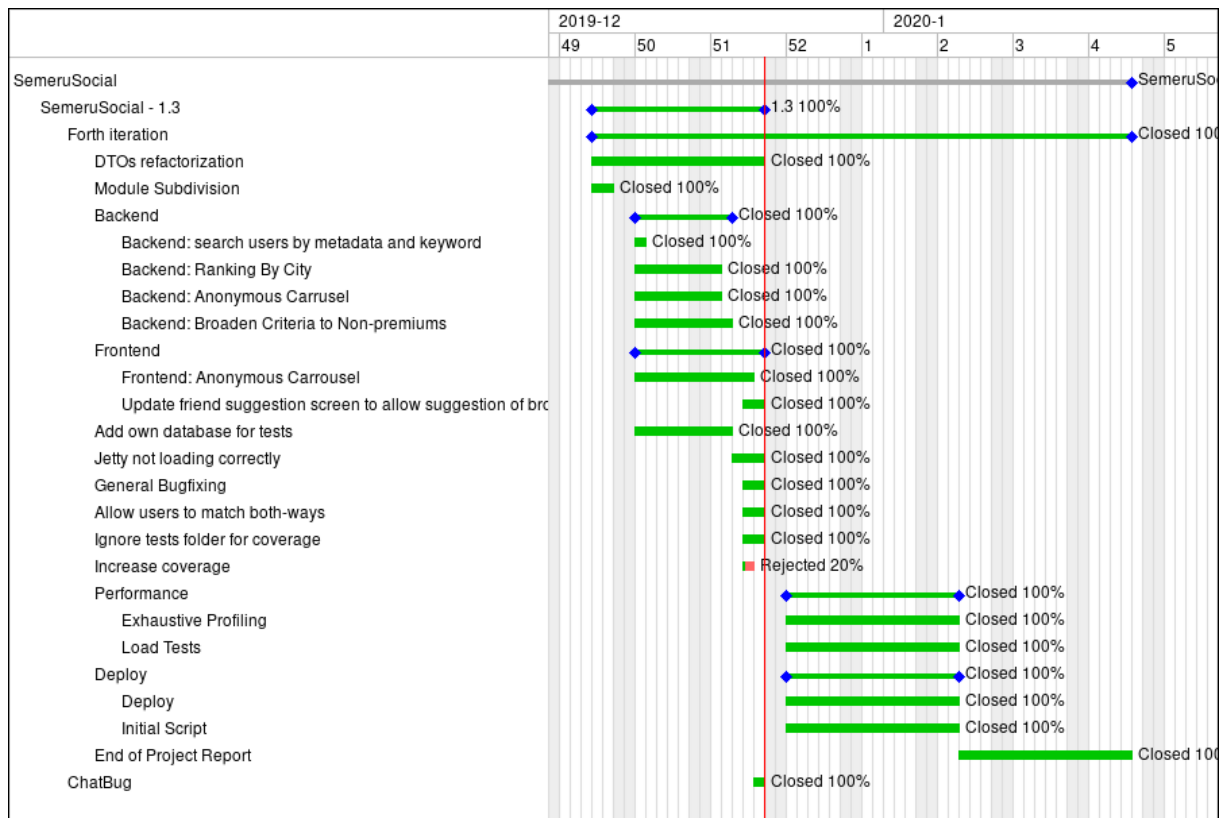
Estado final: Entregado.

	<p>Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p>SemeruSocial</p>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------	----------------------------

3.4.2 Organización del proyecto y documentación

Tag redmine: [SemeruSocial – 1.3](#)

- No se realizó documentación física de diseño más allá de la división de tareas



3.4.3 Consideraciones (e informe):

HERRAMIENTAS:

JMETER:

Contexto:

Los test se han ejecutado en un ordenador con un i7-8550U y 16GB de RAM que es lo que el equipo ha considerado más se puede aproximar al sistema donde la aplicación sera deployeada.

Se ha creado un plan de pruebas que intenta simular el uso normal de un usuario. Como la aplicación es principalmente una API REST para la simulación se pide primero el HTML de la pagina y después de un tiempo aleatorio que se considera que es el tiempo que el usuario tardaría en procesar la información de la pagina y empezaría a interactuar con esta se procesan las peticiones a la API rest que dicho usuario ejecutaría en una interacción normal. Entonces la secuencia seria la siguiente

1. HTML de signup

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

2. API REST singup, se procesa el json para obtener el JsonWebToken que se utilizara para el resto de peticiones del plan.
3. HTML Sugerencia de Amistad
4. API REST Búsqueda de sugerencia. Se procesa el json para obtener el ID del usuario de la sugerencia para las posteriores peticiones
5. API REST Imágenes de usuario. Peticiones del usuario que esta saliendo ahora mismo en la pantalla de sugerencia
6. API REST Aceptar/Rechazar. Se procesa la petición aleatoriamente en base a un generador bool aleatorio.
7. API REST Realiza una nueva sugerencia de amistad guardando su id para posteriores llamadas
8. Otra vez el paso 5.
9. Paso 6.
10. HTML Propio Perfil
11. API REST Actualización de premium. Se actualiza el estado del premium aleatoriamente en base a un generador de bool aleatorio.
12. HTML Lista de Amigos
13. API REST Recuperar lista de amigos.

Esta batería de pruebas se ha utilizado para las distintas pruebas ejecutadas con ligeras variaciones tanto en el numero de usuarios, ramp-up, bucles, configuración del entorno y sleeps para poder alcanzar los distintos escenarios.

Soak/endurance tests:

Se simularon dos escenarios , variando el numero de usuarios concurrentes de la aplicación, pero en todos los escenarios ejecutarían la batería de pruebas 1000 veces por usuario.

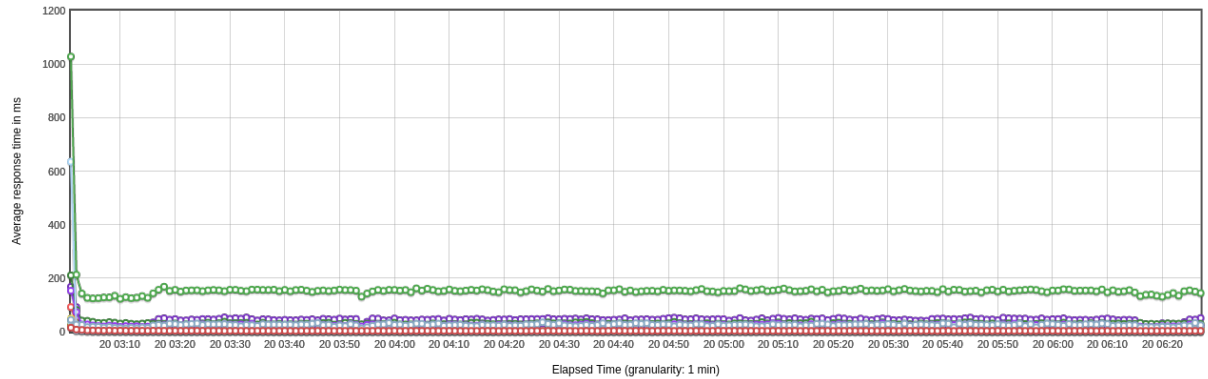
- 10 usuarios por segundo
- 0.03% de errores. (Debido a generación aleatoria de usuarios dando repetidos)

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

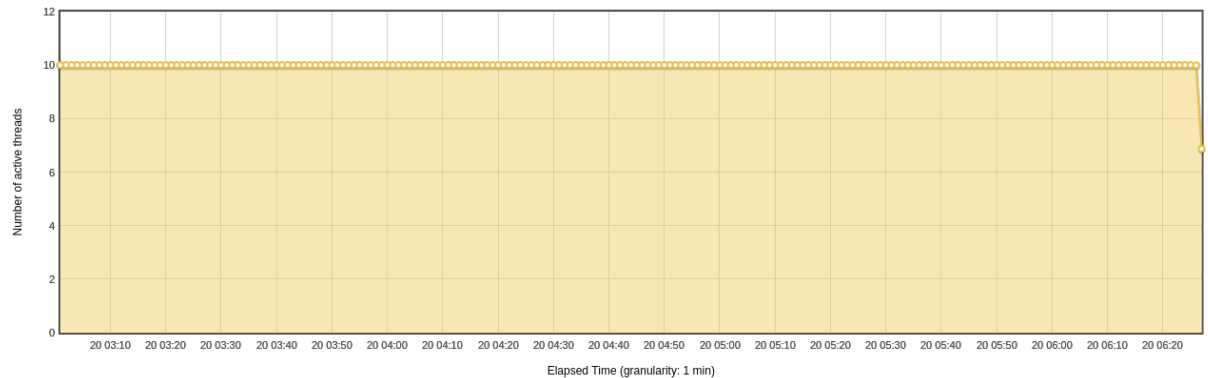


Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas

SemeruSocial

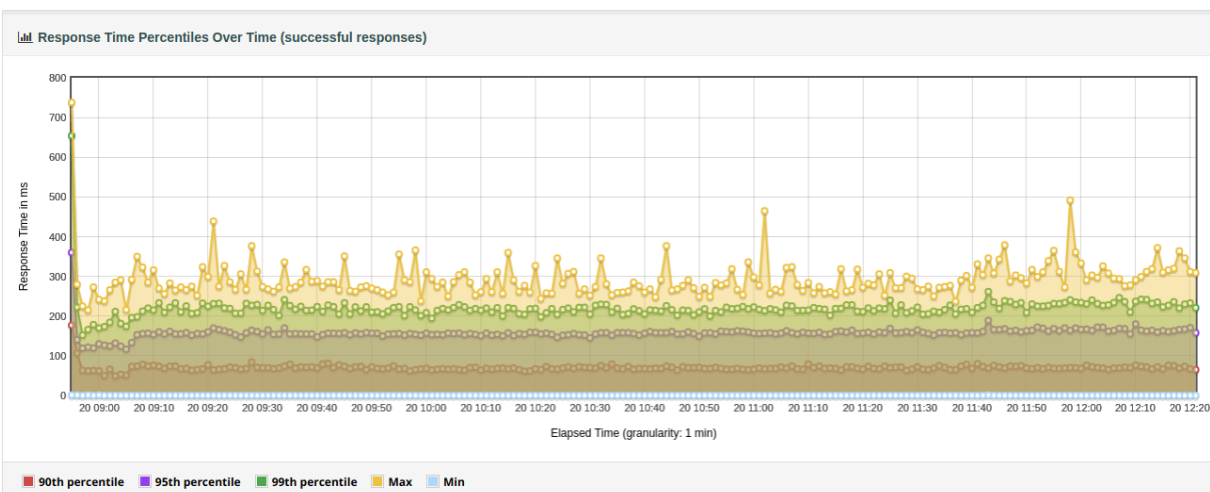
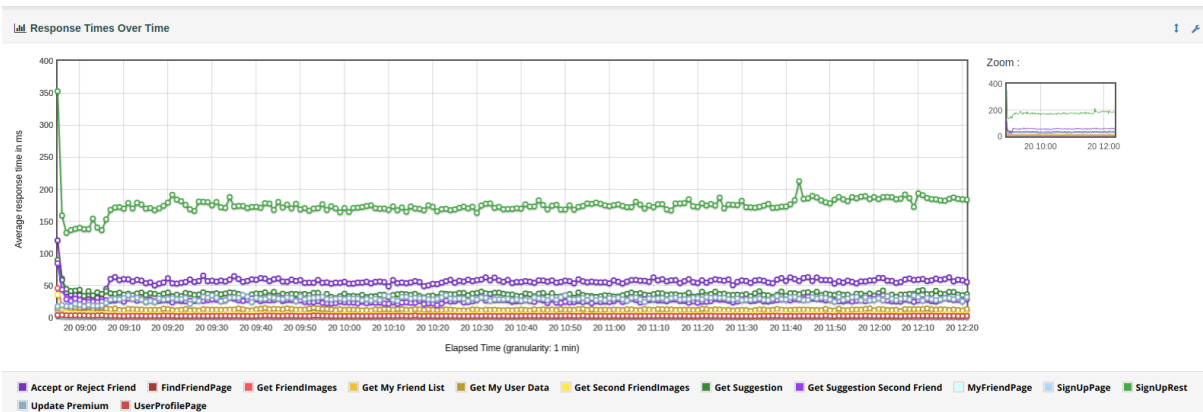


- 3h 26m de ejecución



Se puede observar que el sistema se mantuvo estable durante las tres horas de ejecución dando un resultado exitoso a la prueba

- 25 usuarios por segundo
 - 0.02% de errores (mismos motivos)
 - 3h26m de ejecución



Como conclusiones podemos otra vez mas afirmar que la aplicación response de manera estable subiendo de 10 a 25 usuarios por segundo y que el 90% de las peticiones se ejecutan bastante rápido

Pero podemos ver que la petición que realmente crea los usuarios en la base de datos sufre un incremento con respecto a el caso de 10 usuarios por segundo. Lo que pueda sugerir un problema con las transacciones al hacer inserts en la base de datos.

Stress tests:

Usando misma batería de pruebas que en las pruebas anteriores pero teniendo un throughput muy superior y durante un periodo mucho mas breve.

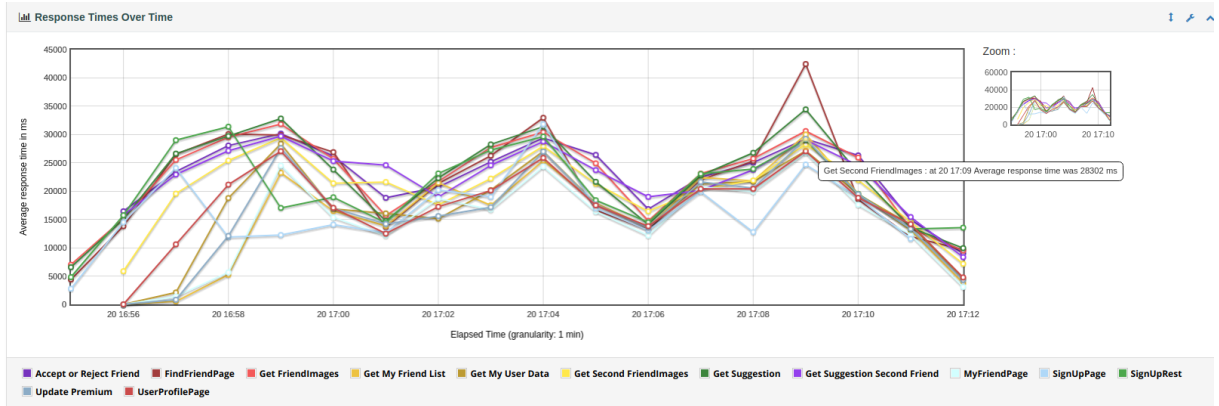
- 15000 user 1 segundo de ramp-up



Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas

SemeruSocial

- 10.59% de errores (*Time outs* y *Too many files open*)
- 17mins de ejecución

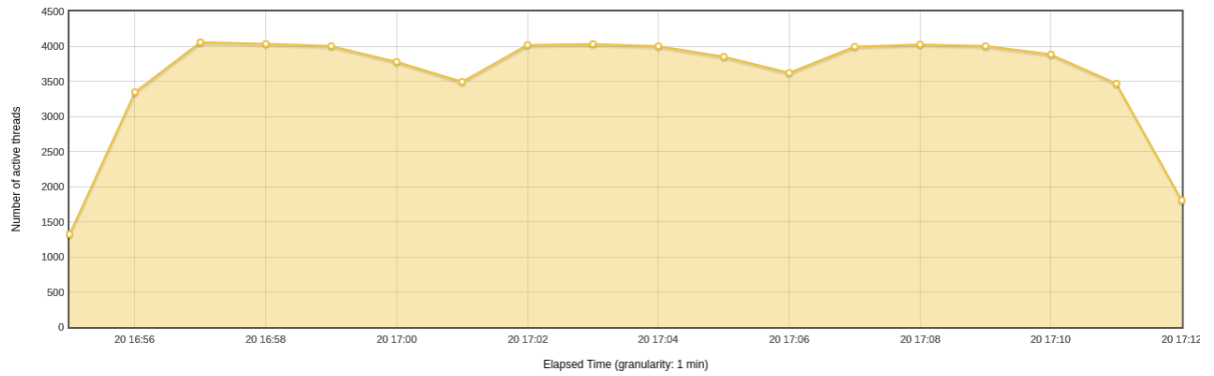


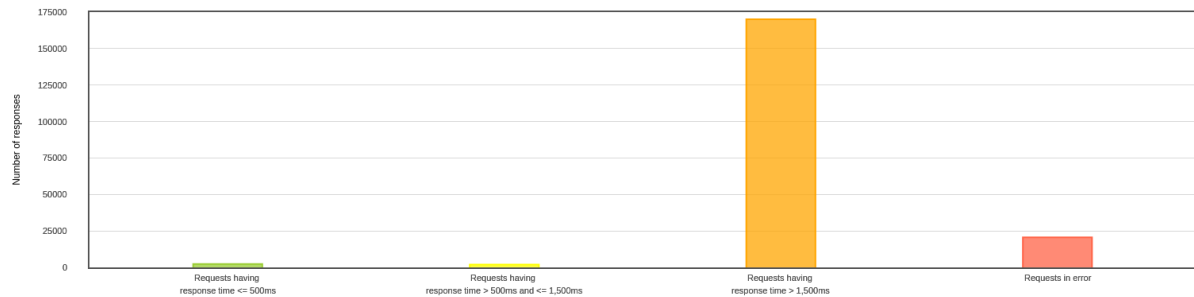


Friendmatch
Informe de Cierre de Proyecto y lecciones aprendidas

SemeruSocial

Active Threads Over Time





Podemos observar que aunque le pongamos 15000 threads sin ramp up el máximo numero de threads alcanzado es siempre de 4000+. Este parece el límite de conexiones abiertas que nuestra entorno acepta para cualquier instante. Y que para este tipo de test la aplicación responde bastante mas lento de lo normal. Pero es capaz de responder a la mayoría.

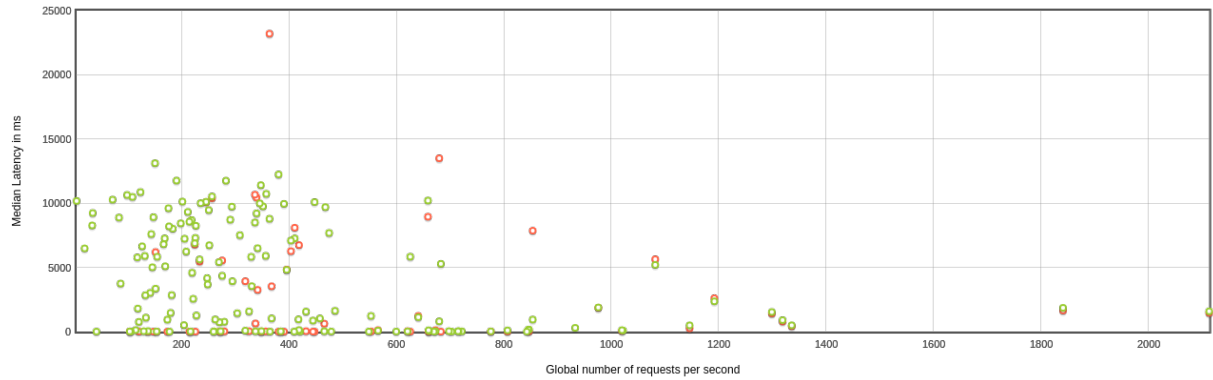
Config tests:

Se han generado distintos tests para las distintas combinaciones que se han usado a lo largo de toda la practica como Gestores de Base (h2,mysql) y Contenedores de aplicaciones(jetty,tomcat9)

Escenario JETTY vs TOMCAT9

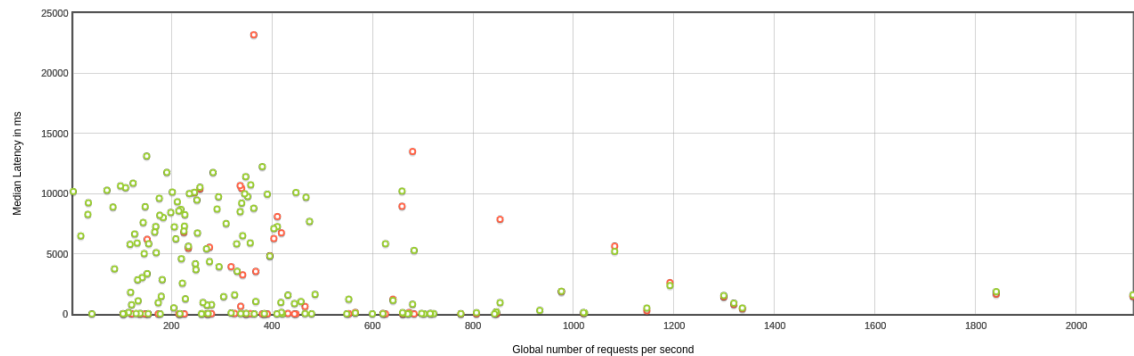
Tomcat

- 5000 user 10 segundos de ramp-up
- 0.78% de errores (*Too many files open*)
- 2 mins de ejecución



Jetty

- 5000 user 10 segundos de ramp-up
- 18.07% de errores (*Too many files open*)
- 6 mins de ejecución



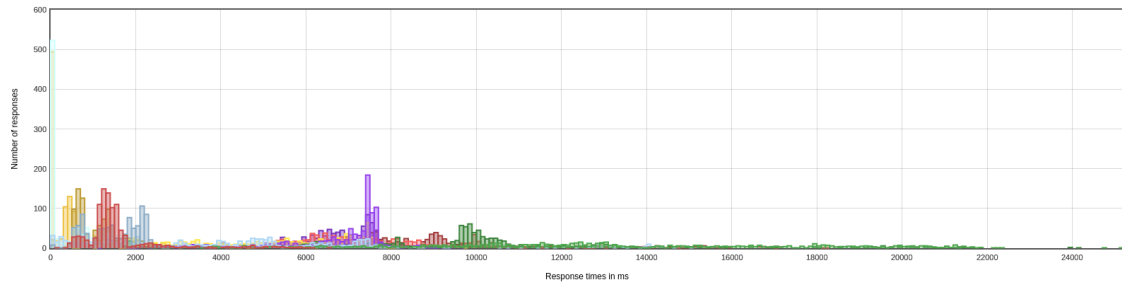
Bajo una situación de media de estrés detectamos una importante diferencia entre jetty y tomcat para nuestra aplicación.

Mas errores para jetty y mucho mas tiempo / request

H2 vs MySQL

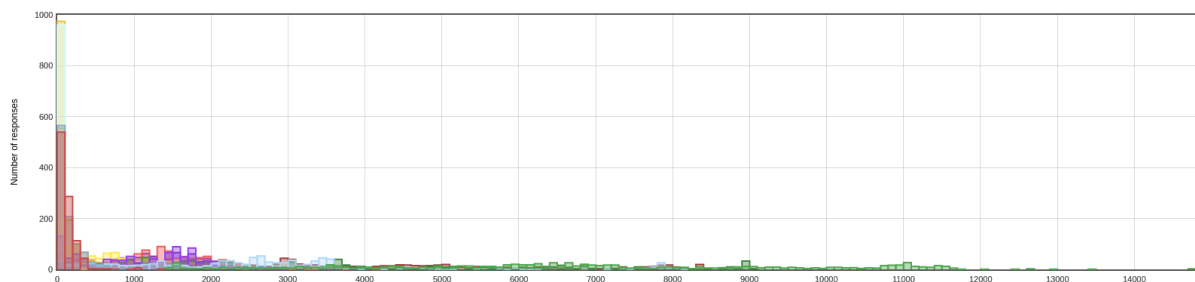
MySQL

- 1000 user 5 segundos de ramp-up
- 0% de errores
- 2 mins de ejecución



H2

- 1000 user 5 segundos de ramp-up
- 0% de errores
- 1 min de ejecución

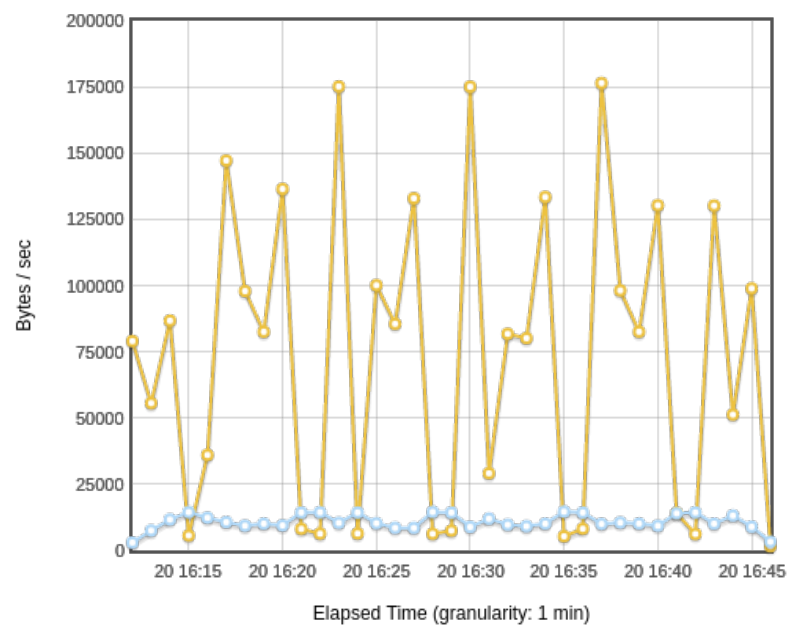
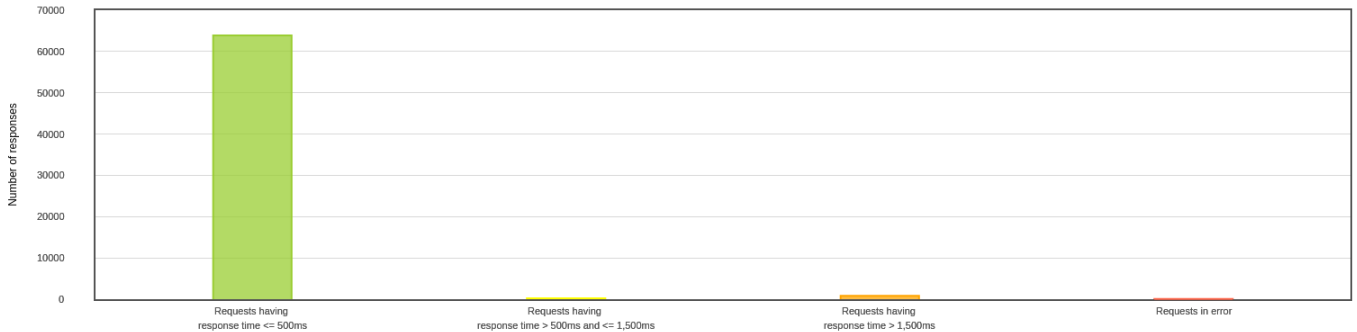


H2 para nuestro sistema parece que rinde mejor que MySQL con muchas mas respuestas que terminan en un umbral de tiempo menor.

Spike test:

Mucho throughput de usuarios , poco ramp-up, muchos bucles de ejecución y un sleep al final de cada thread para simular un descenso de usuario de la aplicación.

- 1000 user 5 segundos de ramp-up
- 0.1% de errores (*Generación de usuarios aleatorios*)
- 34 mins de ejecución



Podemos observar los picos en la información recibida y vemos que ante dichos picos nuestra aplicación mantiene un muy elevado porcentaje de peticiones satisfactorias en bajo tiempo.

Conclusiones:

En la máquina testada se ha encontrado un límite superior de 4000+ conexiones simultaneas para la que la aplicación deja de aceptar más conexiones entrantes.

A mayores no se ha detectado ningún fallo tras una larga ejecución con una carga

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

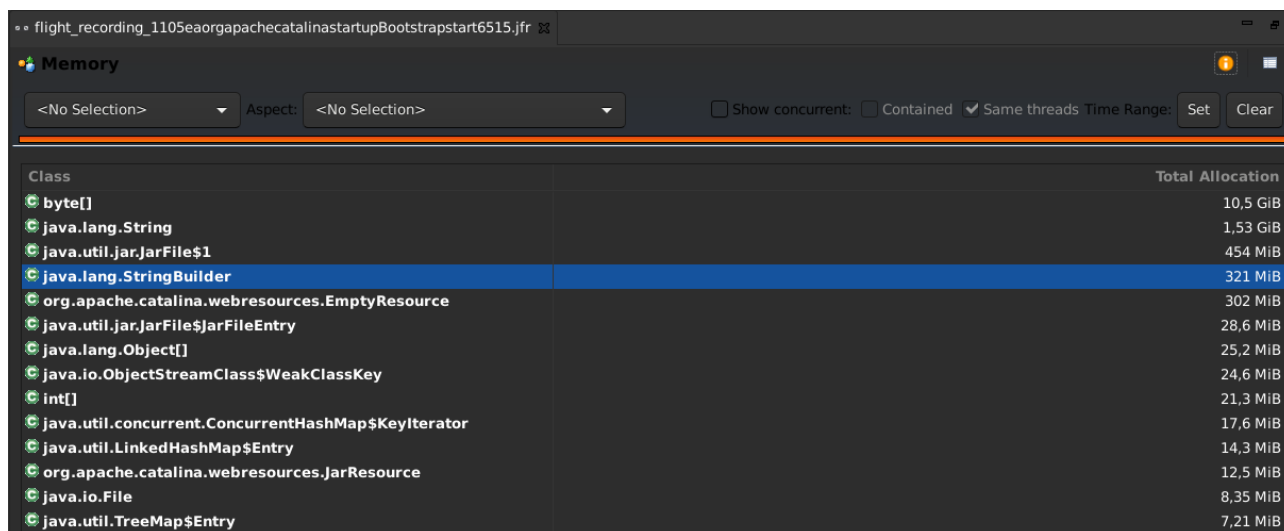
considerable por lo que no se considera que haya *memory leaks* importantes o significativos lo que implica que la aplicación pueda estar un tiempo importante en funcionamiento sin producir muchos errores.

Con respecto a las configuraciones del entorno se ha confirmado que para nuestra configuración tomcat funcionaría mejor que jetty y que la base de datos en memoria ejecuta más rápido que una tradicional como MySQL, eso sí, perdiendo la persistencia en caso de fallo.

Profiling:

Se realizó un análisis de rendimiento de la aplicación. Para llevar a cabo el análisis, se ejecuto la aplicación sobre un Tomcat 9, utilizando MySQL como base de datos. Durante la prueba se sometió a la aplicación a una carga constante de 25 usuarios por segundo, y se inició un flight recording de la aplicación en ejecución. En este análisis se atendió a los siguientes apartados:

- Memoria: Para empezar, el informe general obtenido nos mostró que no encontró memory leaks. Sin embargo, para estudiar el uso de memoria de nuestra aplicación, llevamos a cabo un análisis de los distintos objetos que se crearon. Encontramos así que buena parte de nuestro uso de memoria se realizaba en arrays de bytes (10.5 GB fueron reservados) y en Strings (1.53 GB).



Class	Total Allocation
byte[]	10,5 GiB
java.lang.String	1,53 GiB
java.util.jar.JarFile\$1	454 MiB
java.lang.StringBuilder	321 MiB
org.apache.catalina.webresources.EmptyResource	302 MiB
java.util.jar.JarFile\$JarFileEntry	28,6 MiB
java.lang.Object[]	25,2 MiB
java.io.ObjectStreamClass\$WeakClassKey	24,6 MiB
int[]	21,3 MiB
java.util.concurrent.ConcurrentHashMap\$KeyIterator	17,6 MiB
java.util.LinkedHashMap\$Entry	14,3 MiB
org.apache.catalina.webresources.JarResource	12,5 MiB
java.io.File	8,35 MiB
java.util.TreeMap\$Entry	7,21 MiB

Además, Mission Control nos indicó que parte de la información utilizada en los arrays eran en realidad memoria de los Strings que se estaba utilizando en una técnica de optimización llamada “String deduplication”.

✓ 15	String Deduplication	Approximately 26 % of all allocations were of internal arrays in strings (byte[] for this JDK version). The heap is around 4 % full. There is likely no big benefit from enabling string deduplication.
------	----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas	SemeruSocial
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------	---------------------

De todo esto podemos extraer que buena parte de nuestro uso de memoria se destina a Strings y, por lo tanto, en caso de necesitar obtener una reducción en el uso de memoria, podríamos estudiar no utilizar tantas veces esta clase, o sustituirla por otra que utilice una codificación menos costosa en memoria.

- Uso de CPU:

Respecto a nuestro uso del procesador encontramos que en general no era un cuello de botella en nuestra aplicación. No se detectaron métodos cuya optimización pudiera acelerar especialmente la aplicación, ni tampoco no nos afectó gravemente el recolector de basura:

✓ 1	GC Pauses	Application efficiency was not highly affected by GC pauses.
✓ 0	GC Pause Peak Duration	The longest GC pause was 12,218 ms.

Conclusiones:

Confirmamos las conclusiones de los test de rendimiento y aumentamos más aun la confianza de la ausencia de *memory leaks*.

Deploy:

Para deployar se ha optado por la opción por defecto de dejar que el contexto de los módulos se definan a partir del nombre de los wars, para lo cual les asignamos unos específicos en el pom a la hora de buildear. Para detectar los wars simplemente se añadió la configuración a jenkins para un servidor tomcat en la url dada por el profesorado.

3.4.4 Desviación del proyecto

Stimated time 70.75 hours

Spent time 60.59 hours

Entregado en tiempo.

	<p>Friendmatch Informe de Cierre de Proyecto y lecciones aprendidas</p>	<p>SemeruSocial</p>
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------	----------------------------

4 ASPECTOS RELEVANTES

No aplica.