Capacitación Inicial 2017

Entrega Diseño del modelo "Ratatoullie"

Documento entrega

Grupo 6

Adnan Ferrer

Natanael Nufrio

Elizabeth Minchaca

Alcance

- Un Usuario podrá seguir y ser seguido por otros usuarios, por lo cual tendrá lista de sus seguidores y otra lista de quienes él sigue.
- Un Usuario podrá dejar de seguir a alguien.
- Tanto Usuarios y Restaurantes tendrán una ubicación.
- Un Usuario sera unico por su mail y un restaurante por su nombre y ubicación.
- Usuario Responsable puede tener a cargo muchos restaurantes y mínimo uno.
- Para el ranking de usuarios se aplicó el Patrón State ya que el estado (ranking) puede cambiar de acuerdo a distintas condiciones en tiempo de ejecución.
- Se modela la plataforma (class Ratatouille) usando el Patrón Singleton, que será única y proporcionará un acceso global a ella.
- Se mantendrá el historial de menús con una fecha inicial y final por cada menú y una lista de menús en cada restaurante.
- La fecha inicial es asignada por el sistema cuando se agrega un menú al restaurante siendo este el nuevo menú actual, y se asigna la fecha final al menú que era actual hasta dicho momento.
- Se decidió que tanto los menús como restaurantes podrán ser comentados, para esto se definió una interfaz Commented, de la cual Menú y Restaurante deberán implementar.
- Se podrá comentar solo el menú actual.
- Todo comentario tiene un voto.
- Los comentarios no se responden, van uno detrás del otro ordenados por fecha.
- Cada Restaurant tendrá una Categoría y cada categoría tendrá una lista de Beneficios. Las categorías Implementadas son:
    - Popular(Beneficio:TOP,Votación Total mayor a 10)
    - Neutral(Sin beneficios,Votación Total entre -10 y 10)

- No Popular(Beneficio: TYPOGRAPHY, Votación Total menor a -10).

La votación total se calcula del total de los comentarios del Restaurante incluyendo sus menús, votos positivos suman y negativos restan.

- Un usuario podrá ser normal o responsable, y no podrá cambiar de normal a responsable o viceversa.
- Un usuario podrá deshabilitar su cuenta, si es un usuario normal no podrá comentar su perfil sólo habilitarlo, si es responsable tampoco podrá modificar sus restaurantes. Cuando está deshabilitado sus comentarios se verán como anónimo.
- Un usuario podrá recomendar un menú a un usuario específico o a todos sus seguidores.
- Se pueden listar los usuario con más comentarios y la cantidad de usuarios por Ranking.
- Búsquedas posibles de restaurantes:
  - Por cantidad minima de un tipo determinado de votos.
  - Por ubicación específica y una distancia determinada.
  - Restaurantes con mayor cantidad de comentarios en un rango de fechas dados.
  - Por nombre del restaurante.

Tecnologías usadas:

- **Maven:** es una herramienta de software para la gestión y construcción de proyectos Java, tiene un modelo de configuración de construcción simple, basado en un formato XML.Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.
  - Lo usamos para organizar el proyecto y definir las versiones exactas de librerías a usar.
- **Spring:** es un framework de código abierto creado para abordar la complejidad del desarrollo de aplicaciones enterprise. Una de las ventajas importantes es su arquitectura en capas, que le permite seleccionar el componente que desee, mientras que otros se ignoran. Spring es a la vez amplio y modular. Es un framework ideal para el desarrollo impulsado por pruebas. El principal objetivo de Spring es hacer que J2EE sea más fácil de usar y promover una buena práctica de programación. Esto lo hace al habilitar un modelo de programación POJO que sea aplicable en una amplia gama de entornos.
  - Lo usamos para configurar el contexto, el ORM(hibernate).
  - Para crear beans de servicios y datos.
  - Para cargar la base de datos al iniciar el sistema.
  - Para mapear los controladores de las vistas.
  - Para ofrecer una api Rest.

- **AspectJ:** es un lenguaje de programación orientado por aspectos construido como una extensión del lenguaje Java creado en Xerox PARC. Un compilador de AspectJ hace llegar la noción de aspecto hacia el código de máquina virtual implementando así una noción de relación. Los aspectos en sí se escriben en Java extendido generando un archivo java o compilado con código de máquina compatible con el generado por los compiladores de Java.
    - Usamos aspectos para configurar el ambiente transaccional a nivel de servicios y para realizar el log del sistema también a nivel de servicios.
- **Log4j:** es una biblioteca open source desarrollada en Java por la Apache Software Foundation que permite a los desarrolladores de software escribir mensajes de registro, cuyo propósito es dejar constancia de una determinada transacción en tiempo de ejecución. Log4j permite filtrar los mensajes en función de su importancia. La configuración de salida y granularidad de los mensajes es realizada en tiempo de ejecución mediante el uso de archivos de configuración externos.
    - En nuestro sistema la usamos como herramienta de log, configurada en el archivo log4j.xml.
    - Se implementó un log a nivel de servicios, y se loguea por consola y en un archivo.
- **Hibernate:** es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.
    - Lo usamos en nuestro modelo para mapeo de las clases a la base de datos usando anotaciones y para las consultas a la base.
- **C3P0:** es una biblioteca para el manejo de una colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones.
    - Configuramos hibernate para que C3PO administre sus conexiones a la base.
- **JavaServer Pages(JSP):** es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. **JSP** es similar a PHP, pero usa el lenguaje de programación Java.

- ➢ La usamos para el desarrollo de las vistas.

- ❖ **AngularJS o AngularJS 1:** es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.
  - ➢ Usado como framework SPA
- ❖ **Bootstrap:** es un framework o conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.
  - ➢ Usado para dar estilo en las vista, también se usó css.
- ❖ **Bootstrap-tables.js:** plugin de boostrap para desarrollo de tablas.
- ❖ **jQuery:** es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es la biblioteca de JavaScript más utilizada.
  - ➢ Usada para simplificar el desarrollo de los scripts.
- ❖ **Sweetalert 2:** reemplaza los alerts por defecto de Javascript por unas cajas de diálogo más coloridas y animadas. Hay cinco tipos diferentes de mensajes, un montón de opciones de personalización y todos los métodos que necesites. SweetAlert2 es compatible con todos los navegadores modernos mensajes de alerta en las vistas.
  - ➢ Usada para mostrar notificaciones al usuario.
- ❖ **Google Maps Api:** es un servicio gratuito que nos ofrece Google con mapas desplazables del mundo entero, fotos satelitales, la ruta más corta entre diferentes ubicaciones y muchas características interesantes.
  - ➢ Usada para establecer la ubicación de los usuarios y los restaurantes.
  - ➢ Para realizar búsquedas por ubicación.
  - ➢ Para convertir las coordenadas de usuarios y restaurantes en direcciones.

Problemas/Soluciones

1. Cada usuario y restaurante debía tener una ubicación en un mapa, y luego se debían hacer operaciones como calcular distancia entre restaurantes, buscar restaurantes en un radio.
    - Para solucionar dicho problema optamos por usar la api de Google Maps la cual debimos aprender y usar en las funcionalidades que era necesario.
2. Para el desarrollo de las vistas del sitio web se debía elegir entre realizar un desarrollo estándar usando JSP o realizar un desarrollo SPA.
    - Optamos por realizar un desarrollo SPA para lo que debimos aprender a usar Angular 1.
3. Al momento de mapear las clases a persistir se debía elegir entre mapearlos con archivos .xml, usar anotaciones JPA o anotaciones de Hibernate. Se optó por usar anotaciones JPA y anotaciones Hibernate donde no se pudo usar JPA.
4. Había que definir un ambiente transaccional, se podía definir usando anotaciones a nivel de clase o a nivel de método; se optó por usar aspectos y definir el ambiente desde la configuración de spring.
5. Se debió hacer un log del sistema y en lugar de agregar las instrucciones dentro de cada método, se volvió a usar aspectos para logu0ear el acceso, salida y salida errónea a los métodos.

Conclusiones

El proyecto ha logrado cumplir todos los objetivos básicos que se habían propuesto y respetando las especificaciones planteadas.

Esto se realizado utilizando una estructura en capas que permite una fácil extensibilidad de la aplicación, lo que ha permitido que los integrantes del grupo colaboren con el proyecto e implementen partes distintas sin afectar al lo desarrollado por el resto. Esta estructura en capas también permite que los errores sean más fáciles de localizar.

Por otra parte, del lado del cliente se ha creado una agradable a la par que simple interfaz y que da la sensación de rapidez al usar un framework SPA y no necesitar recargar toda la página al navegar por el sitio.

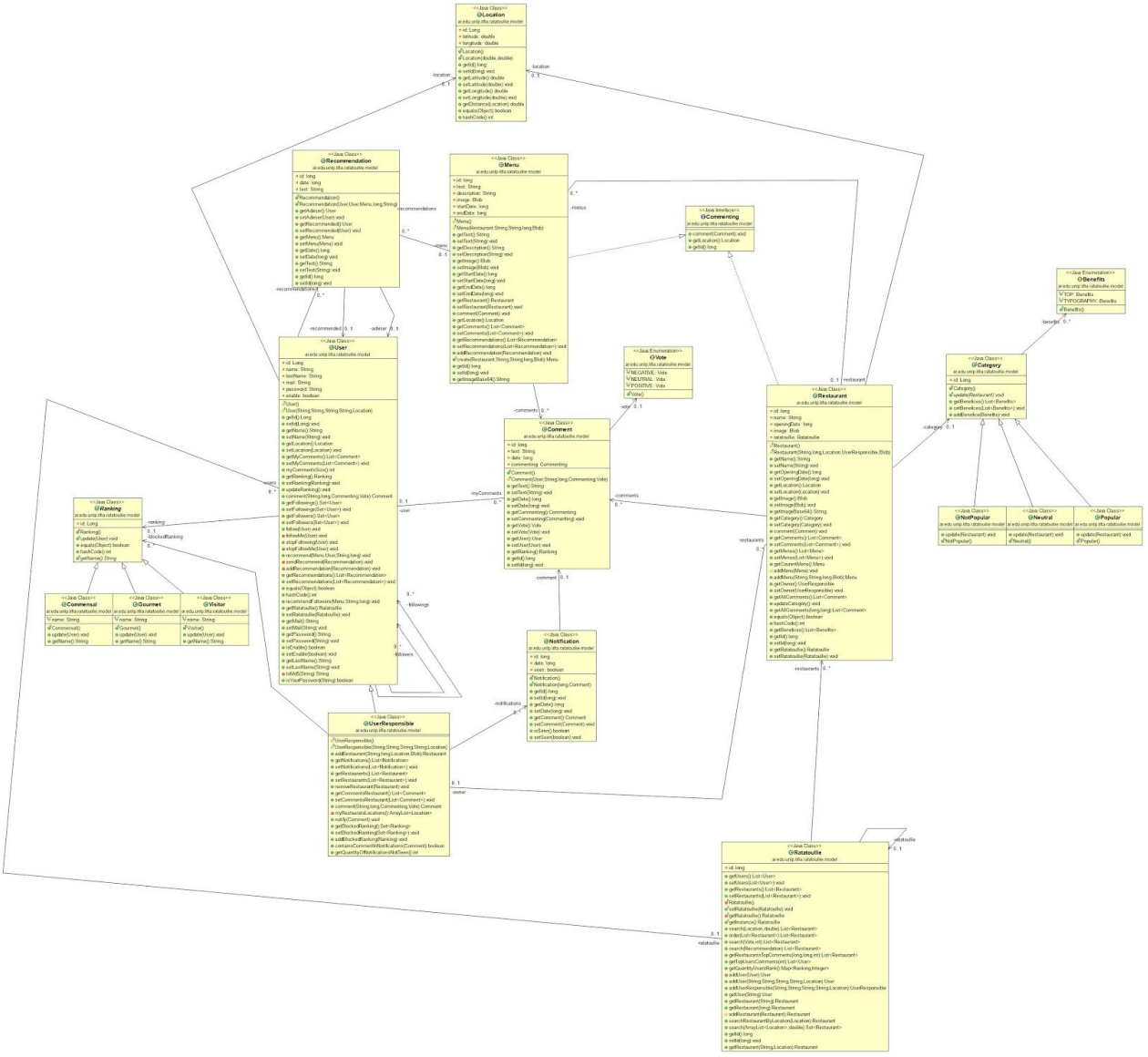
El resultado ha sido probado en distintos navegadores modernos con satisfactorio resultado, gracias a que ha sido desarrollado empleando estándares y tecnologías abiertas.

En cuanto a los desarrolladores nos ha servido para aprender nuevas tecnologías como Spring, Hibernate y apis como la de Google Maps.

Extensiones/ Mejoras

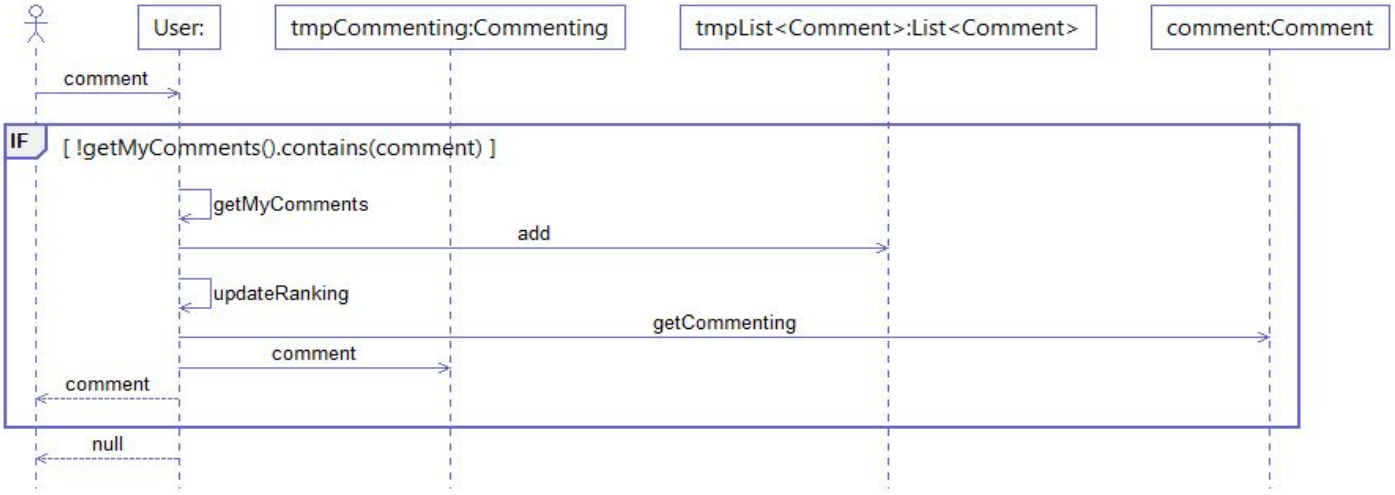
1. Sería normal pensar que un usuario común quisiera agregar un restaurante propio en algún momento, en la versión actual del sistema debería registrarse con una nueva cuenta, una mejora posible sería que una cuenta de usuario común pueda pasar a ser usuario responsable y viseversa.
2. Cada restaurante tiene un historial de menús, actualmente solo se puede comentar el menú que está en vigencia, se podría mejorar agregando la opción de comentar cualquier menú.
3. Los comentarios se muestran uno detrás del otro ordenados por la fecha en que fueron realizados, una mejora seria como es común en las redes sociales se podría responder un comentario.
    a. Como cada comentario tiene un voto de quien lo realizo, tambien se podria agregar votos de otros usuarios, como es normal en las redes sociales.
4. Cada usuario tiene una lista de seguidores, una mejora a realizar sería poder bloquear a los seguidores como es normal en casi todas las redes sociales.
5. Para la búsqueda por ubicación se toma la ubicación del navegador y si esta falla se establece como ubicación el centro de La Llata, como cada usuario establece una ubicación al registrarse podría tomarse esta para establecer la ubicación.
6. Al momento de ver la foto del restaurant se usa el Carousel de Bootstrap, como mejora a futuro se podría tener una lista de fotos o pasar al siguiente o anterior restaurant .
7. Cuando se lee una notificación se muestra un botón que vuelve al listado de notificaciones, como mejora se podría agregar 2 botones: para pasar a leer el siguiente y la anterior notificación.
8. En el presente no se disponen funcionalidades de eliminar registros en ningún caso. Como implementación futura podría hacerse bajas lógicas y mantener el historial de datos eliminados.

Diagrama de clases

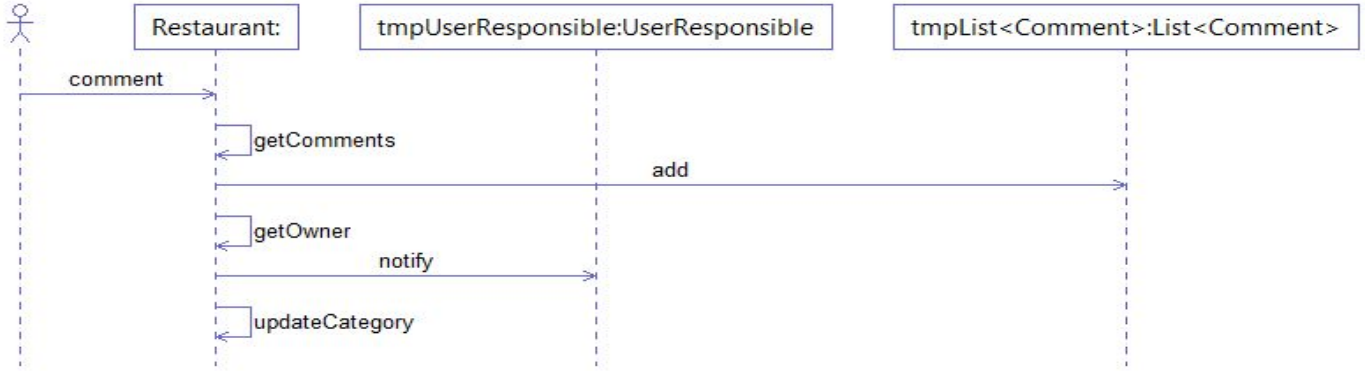


Diagrama/s de secuencia

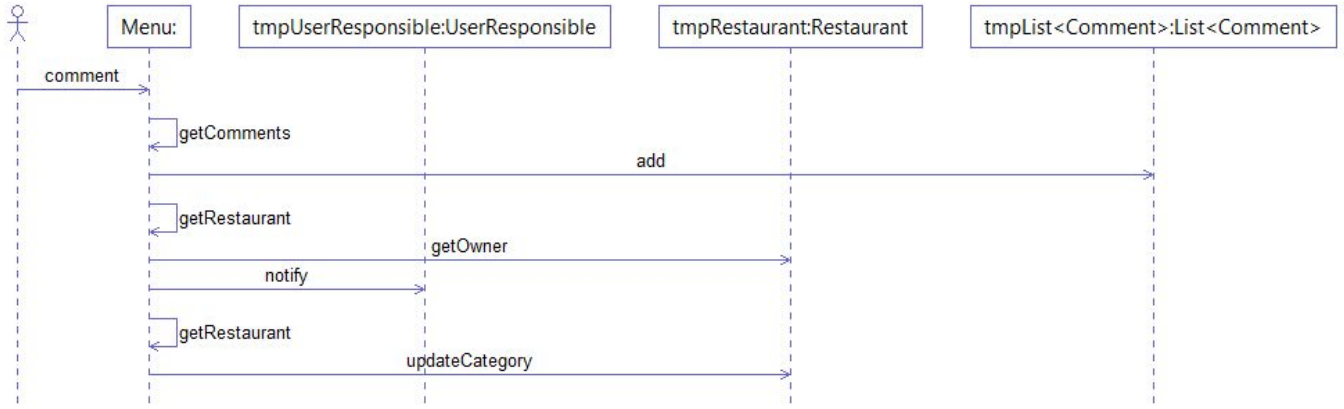
1. User.coment(): A continuación se muestra la realización de un comentario por un User(Usuario Normal) a la interface Commenting, posteriormente se mostrara las dos implementaciones de Commenting(Menu y Restaurant), un User, verifica q no contenga el comentario, luego lo agrega a su lista de comentarios, actualiza el ranking y realiza el comentario sobre el Commenting.



2. Restaurant.comment():agrega el comentario a su lista de comentarios, notifica al Owner(Usuario Responsable del restaurante) y actualiza su Category(Categoría).



3. Menu.comment():agrega el comentario a su lista de comentarios, notifica al Owner(Usuario Responsable del restaurante) y actualiza la Category(Categoría) del restaurante al que pertenece.



4. UserResponsible.comment(): un usuario responsable verifica que ninguno de sus restaurantes este próximo a quien debe comentar y luego ejecuta el comentar de la clase padre.