Práctica Obligatoria: *Evolución y Adaptación de Software*

Curso 2019/2020: durante la Gran Cuarentena

Contenido de la Práctica

Los próximos puntos resumen brevemente el enfoque general de la práctica, que ya se había comentado en clase, y que también se ha descrito someramente en mensajes anteriores en los foros de la asignatura. En primer lugar se describe la secuencia de pasos que describen la práctica (A), y luego se resume brevemente su objetivo (B). Las normas de entrega, necesariamente sujetas a revisión, se describen en la sección final (C).

A. Definición de la Práctica

---

1. Se ha facilitado, en un repositorio público en GitHub, el código fuente de una aplicación en Java. [1]
    - https://github.com/Sw-Evolution/JRoar
    - La aplicación es un servidor para *streaming* de audio, basada en el formato multimedia Ogg (concretamente, en su parte de audio, Vorbis). El código original es un desarrollo de JCraft, liberado bajo licencia GPL.

      (*Más detalles, en la sección B.*)
2. Si no lo ha hecho ya, cada grupo de alumnos deberá facilitar a los profesores, mediante correo enviado por el Aula Virtual, la lista de los integrantes de cada grupo de prácticas y sus usuarios de Github.
    - Los profesores crearán (como ya se ha hecho en muchos casos) un grupo y repositorio Github para esos usuarios (dentro de la organización https://github.com/Sw-Evolution). Estos repositorios son *privados* y sólo son accesibles por los miembros de cada grupo y por los profesores.
    - Los usuarios Github correspondientes recibirán automáticamente su invitación para participar como miembros de ese grupo, y deberán aceptarla.
3. Cada grupo deberá clonar el código del repositorio público en su repositorio privado. **Todo el trabajo de desarrollo deberá hacerse bajo control de versiones (git) en este repositorio**, de modo que pueda ser evaluado posteriormente por los profesores.

   Dado un hipotético grupo 0 (20E00), cuyo repositorio sería https://github.com/Sw-Evolution/20E00, una posible secuencia de comandos que realizarían adecuadamente esta clonación sería la siguiente:

   ```
   $ git clone git://github.com/Sw-Evolution/JRoar.git
   $ cd JRoar
   $ git remote remove origin
   $ git remote add origin https://github.com/Sw-Evolution/20E00.git
   $ git push -u origin master
   ```
4. La práctica en sí consta de dos partes:
    1. **Mantenimiento**. El código aportado es antiguo (la última versión es de 2002), y tiene características que han quedado obsoletas por evolución del lenguaje de programación (Java), así como algunas otras que no se consideran "modernas" (aunque todavía "funcionan"). Por otra parte, es un programa concebido como "puro Java", por lo que resulta una implementación bastante "limpia", a pesar de su antigüedad. Cada grupo habrá de adaptar el código para que funcione correctamente en un entorno de programación actual (de su elección), haciendo *todas las modificaciones que considere necesarias*.
        - No se pide que estas modificaciones sean "mínimas" - cada uno puede modificar todo el código que estime conveniente, incluso cambiando la forma en la que se implementen estructuras de datos, si es su intención.
        - Se pueden eliminar todas las partes que se consideren accesorias, como (*por ejemplo*) las que se encargan del (difunto) protocolo PeerCast o las propias del cliente JOrbisPlayer. No deberán eliminarse cuando se opte por **utilizarlas** explícitamente, esto es, cuando se decida que serán parte de las ampliaciones propuestas por el grupo. Se valorará especialmente el esfuerzo realizado para eliminar "código inútil" del sistema.
        - Se *permite*, pero **no** se exige, el uso de herramientas de análisis estático de código (v.g. SonarQube), aunque se

recomienda usar sus conclusiones con mesura. No todo lo que dice un analizador es, automáticamente, un *code smell* que debe ser corregido. Debe considerarse una *ayuda*, no una lista exhaustiva.

[**Nota**: *no se habilitarán permisos especiales en el repositorio* para permitir el uso de SonarCloud. Quien desee usarlo, debería plantearlo de otra manera.]

- Por supuesto, se debe tener especial cuidado en que *todas* estas supresiones no modifiquen ni comprometan la utilidad del sistema, o provoquen después comportamientos inesperados.

2. **Evolución**. Cada grupo habrá de **proponer** *e implementar* una **extensión/ampliación/modificación** de la aplicación original, obteniendo una nueva versión del servidor JRoar (o *elementos de su ecosistema*) que implemente nuevas funcionalidades, o las modifique (o haga cambios significativos en su estructura u organización interna). En general, todas las modificaciones que se planteen deben ser comentadas en cada grupo de prácticas y deben ser previamente "aprobadas" (para comprobar su suficiencia o limitar un exceso de carga) por los profesores de la asignatura, mediante consulta previa (interactiva o por correo).

   - La parte relativa a evolución, por tanto, consta de dos partes: la *propuesta* de modificación/ampliación, y el *desarrollo* de dicha modificación/ampliación.

   - Se tendrán en cuenta especialmente los aspectos de *evolución y diseño*. Los relativos a programación y proceso de desarrollo (ambos accesibles a partir del sistema de control de versiones) se podrán tener también en cuenta, pero en cualquier caso se consideran *secundarios*.

   - Existen múltiples ejemplos de servidores de *streaming* similares más modernos, aunque también, típicamente, más "pesados". Como inspiración, se proporciona también un enlace al repositorio de uno de estos sistemas, Libresonic, la versión libre de Subsonic (https://github.com/Sw-Evolution/libresonic), que es un simple *fork* del repositorio público que contiene el código del *software* del mismo nombre.

5. Originalmente, la intención era realizar la entrega de la práctica **el mismo día del examen de teoría**, planificado originalmente para el 7 de mayo de 2020. Obviamente, en las circunstancias actuales, **ya se ha descartado** mantener esta fecha, ya que nuestro retorno a una evaluación convencional depende de un cúmulo de circunstancias externas. Por todo ello, nos referiremos a la fecha de entrega, de manera general, como **el Día E**.

   (*Más detalles, en la sección C.*)

B. Objetivo global de la práctica.

JRoar, en su concepción original, es un programa que se inspira en la existencia de Icecast, a su vez concebido como una evolución "libre" del protocolo privativo llamado Shoutcast.

- Icecast 2 puede descargarse de su web oficial: http://icecast.org/.
- La documentación disponible en esa web puede servir de ayuda en el desarrollo de la práctica.

La idea es sencilla: Icecast es un servidor de *streaming* multimedia (aunque aquí nos centramos únicamente en el soporte de audio) que utiliza un protocolo HTTP como base subyacente, y el formato multimedia Ogg, concretamente su versión de audio, Vorbis, para funcionar como una "emisora" en Internet.

Es decir, Icecast (e igualmente, JRoar) "reproduce" un fichero Ogg/Vorbis de audio en una URL HTTP determinada. Cualquier programa capaz de reproducir ficheros Ogg desde una fuente remota puede conectarse a esa URL y escuchar, en directo, la transmisión que el servidor está haciendo en ese momento.

- Por ejemplo, una emisora activa es **XRM Radio**: http://stream.xrm.fm/xrm-alt.ogg (su canal *alternativo*)

- Se proporcionará una breve lista con varias emisoras de este tipo, ideales para hacer pruebas.

- **No** se trata de *falso streaming* como el que se hace en otros casos: el reproductor no se "descarga" un fichero de audio (ogg) y lo reproduce en local; sino que llena un pequeño buffer de memoria con unos pocos segundos de audio y va reproduciéndolos en directo, mientras sigue recibiendo el resto de la emisión.

- Hay multitud de reproductores capaces de recibir estas emisiones. Dos ejemplos clásicos son:

  - VideoLAN Player (**VLC**), descargable en https://www.videolan.org/vlc/index.html, *multiplataforma*.
  - **Foobar 2000**, descargable en https://www.foobar2000.org/, con versiones para Windows, Android e iOs.

Icecast no es realmente el que está reproduciendo el audio; sino que es sólo el software que lo *publica* en Internet. Es, literalmente, un *servidor de streaming*, que se encarga de implementar tanto el servidor HTTP que le da soporte como de "desplegar" el *stream* de audio (y/o vídeo) sobre dicho soporte, utilizando para ello las características propias del formato multimedia Ogg (y su codificación de audio, Vorbis). Soporta otros formatos, pero aquí nos centraremos en ése (que es, además, el formato nativo de audio en Android).

Para que Icecast pueda emitir un *stream*, tiene que haber un segundo software, un *cliente origen* (*source client*) que lee un fichero Vorbis (sea, por ejemplo, `The_River.ogg`) y se lo envía a Icecast. Es decir, este cliente origen lee el fichero como si fuera un reproductor de audio, pero en lugar de emitirlo por los altavoces, se conecta al servidor Icecast y le transmite el audio tal como se reproduciría.

Icecast (y también JRoar, que es idéntico en este sentido) recibe este *stream* del cliente origen, y lo emite en su URL pública, de modo que cualquier reproductor que se conecte a ésta será capaz de reproducirlo.

Hay una multitud de clientes fuente compatibles con Icecast (y con JRoar).

- El más famoso es **IceS** https://icecast.org/ices/. IceS se encarga, por tanto, de leer un fichero Vorbis, y mandárselo a Icecast, para que éste lo reproduzca.

Hay un completo listado de clientes de este tipo, para varios sistemas operativos, en la página oficial de aplicaciones compatibles con Icecast: https://icecast.org/apps/.

JRoar es capaz de sustituir tanto a IceS como a Icecast:

- JRoar puede leer un fichero Vorbis y mandárselo a un servidor Icecast, reemplazando a IceS.
- JRoar puede "recibir" un stream desde IceS y publicarlo en una URL dada, reemplazando a Icecast.
- JRoar puede hacer **ambas cosas** a la vez: es decir, leer un fichero Vorbis y mandárselo a un servidor JRoar, y recibir un *stream* de un cliente fuente, y reproducirlo en una URL.

En particular, JRoar es capaz de funcionar como "repetidor", esto es, recibir un *stream* de audio que esté publicado en cierta URL (digamos, http://servidor-lejano.com/emisora.ogg) y publicarlo a su vez en otra URL en la que él está funcionando como servidor (digamos, http://mi-servidor/mi-emisora.ogg). Cualquier reproductor que se conecte a la *segunda* URL reproducirá el mismo audio que emita la primera, acumulando solamente un pequeño retardo.

- Lógicamente, esta operación se puede repetir un número indefinido de veces.

El objetivo que inspira a JRoar es definir una especie de **red P2P de emisoras de radio digital**, en la que cualquiera puede iniciar una emisión (usando JRoar o cualquier cliente fuente), y una emisora JRoar lo publica en una URL (URL1), otra emisora JRoar lo toma de ésta y lo publica en otra URL (URL2), y así indefinidamente. De modo que un oyente sólo tenga que conocer la emisora más cercana (URL*x*) para reproducir la emisión sin sobrecargar al servidor inicial.

En este sentido es similar a una red de radioaficionados, aunque obviamente menos interactivo. Tiene el obvio inconveniente de que si se pierde un solo nodo de la cadena, toda la "rama" de emisoras se pierde; pero sigue siendo un modo curioso de utilizar una emisora en tiempos de cuarentena.

El protocolo PeerCast (http://peercast.sourceforge.net), al que se puede considerar extinto en muchos sentidos, tenía un objetivo similar. Sin embargo, su infraestructura y arquitectura (más allá de utilizar un sistema P2P) son lógicamente diferentes.

Hay muchos más detalles sobre esto. JRoar está ligado a un reproductor de Vorbis, llamado JOrbis, que puede combinarse con él o utilizarse de manera independiente. Véase la documentación del propio JRoar: el README.txt original e incluso su página web, para más detalles.

El **objetivo de esta práctica** es partir de esta idea, y de la estructura original de JRoar, para **modificarla como el grupo de prácticas desee**, sea añadiendo capacidades, suprimiéndolas, combinándolos con otros elementos, etc.

Importante:

- No es es **necesario** que las prácticas hagan realmente una "red de emisoras", es decir: se **puede cambiar de objetivo**. Esto es solo una inspiración. Si a un grupo se le ocurre otro modo de aprovechar la infraestructura, lo puede acometer, por supuesto. Tan solo se requiere la aprobación previa de los profesores de la asignatura para asegurar que el objetivo tiene sentido en el contexto de la asignatura.

- Es una práctica de **reingeniería** y funciona a nivel de **diseño y de programación**, no de despliegue y explotación. No se trata de combinar muchos nodos y muchos *software* emisora/cliente distintos. El objetivo es **diseñar** una arquitectura software e implementar un *software* de acuerdo a la misma, basado en JRoar.

Nota Final: para facilitar la conectividad entre emisoras situadas en lugares físicamente distantes, se permite (e incluso se aconseja) el uso de sistemas de registro de DNS Dinámica (DDNS), tales como DDNS, No-IP, o cualquier otro.

C. Normas finales de entrega

Como se comentó a principio de curso, la entrega de la práctica consiste en el código con *todas* sus versiones (es decir, el repositorio completo de GitHub, con todo su historial) y una **breve** Memoria (por ejemplo, no superior a 10 páginas, aunque esta indicación es **orientativa**) que se enviará a través del Aula Virtual.

Es decir, se entregan **dos** elementos:

1. Se entregará la memoria en el Aula Virtual, en la fecha indicada (**Día E**).
2. Ese mismo día, los profesores clonarán el repositorio GitHub de cada grupo. Se considerará que el contenido de este repositorio (no únicamente la versión final)

De manera detallada:

- Todos los grupos de prácticas tendrán en su repositorio privado (ver sección anterior) la versión final de la práctica, así como todas las versiones anteriores. Como se ha indicado, las versiones intermedias también serán tenidas en cuenta en la evaluación (aunque obviamente se dará más prioridad a la versión final).
    - Se considerará como **versión final** a la última versión del repositorio de cada grupo en GitHub que tenga la **Fecha del Día E o uno anterior**, con la hora de España.
    - No obstante, los repositorios **no se cerrarán** en esta fecha. Las versiones posteriores al Día E **no serán consideradas** de cara a la evaluación en la convocatoria ordinaria, pudiendo acumularse en la extraordinaria.
- La **entrega de la memoria de la práctica** se hará mediante la Tarea de Entrega creada a tal efecto en el Aula virtual (sección de Evaluación).

Cada grupo debe enviar **una única memoria**, que podrá ser entregada por cualquiera de sus miembros como representante de todos ellos. Es conveniente (pero no obligatorio) dejar una copia de la memoria también en el repositorio. En caso de discrepancia, no obstante, se utilizará siempre la versión de Aula Virtual.

Normas básicas sobre la Memoria:

- Debe indicar explícitamente el número del grupo correspondiente en la portada.
- Se valorará explícitamente la **brevedad** y **precisión** de la memoria. A pesar de la indicación de 10 páginas, **no** se indica ni límite máximo ni límite mínimo de páginas, pero se recomienda explícitamente que no sea muy extensa.
- Se puede utilizar texto y diagramas (UML y/o informales), como se desee.
- La memoria ha de constar de **tres partes**:
    1. Primera parte, relativa al *mantenimiento*. Explicar cómo se ha adaptado el código original, y qué elementos se han eliminado. Debe ser **muy** breve y resaltar los aspectos que hayan resaltado más conflictivos (y su solución).
    2. Segunda parte, relativa a la ***propuesta** de evolución*. Debe indicar **claramente** qué se pretendía realizar como extensión, incluso si no se llega a realizar por completo.

       Esta parte es **muy importante**, de cara a evaluar la siguiente.
    3. Tercera parte, relativa a la *implementación* de la propuesta. Se debe indicar claramente qué se ha realizado, así como lo que **no** se haya podido realizar por falta de tiempo u otro motivo, indicando claramente los límites de lo que se pretende evaluar.

       El objetivo no es duplicar el código (esto ya se indica en el repositorio), sino resaltar los **méritos** de la práctica.

Sólo se considerará **presentados en prácticas** a los grupos que envíen la memoria de prácticas por este medio.

- En su defecto (si el grupo tuviera cualquier problema en el momento de la entrega), el grupo tendrá opción a dejar una versión **final**

de la memoria en el repositorio de Github, pero **habrá de indicar este hecho también** por correo electrónico al profesor de teoría, tanto en el aula virtual como a su dirección institucional convencional (carlos.cuesta@urjc.es).

Independientemente del contenido que haya en el repositorio, todos aquellos grupos que **no** hayan entregado la memoria del modo indicado se considerarán automáticamente como **no presentados** en la parte práctica de la asignatura.

---

[1]: Se agradece explícitamente a GitHub el apoyo logístico para la realización de esta práctica, dentro de su programa académico.