Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Iñigo Gascón Royo 685215
Rubén Moreno Jimeno 680882
Guillermo Robles González 604409

Sistemas Legados
Práctica 2
Encapsular el acceso a un mainframe

7º cuatrimestre
Ingeniería Informática
02 de noviembre de 2016,
Zaragoza

Contenido

1. Asegurar que la aplicación funciona correctamente y entendimiento de su funcionamiento

La exploración del sistema Legado se realizó en 3 fases:

1. Durante la primera se usó el programa "x3270" el cual ofrece una TUI al sistema legado trabajado. Con ayuda de la TUI se creó un diagrama sencillo de los distintos pasos involucrados en el uso del sistema (logueo, entrada en "TAREAS.C" y distintas funcionalidades soportadas en "TAREAS.C")
2. Una vez se completó la exploración del sistema de trabajo, se usó la herramienta "s3270", la cual ofrece una funcionalidad similar a x3270, pero en un interfaz de consola ("prompt"). Con ayuda de los diagramas desarrollados en el paso previo, se comprobaron las secuencias de entradas para las distintas tareas de trabajo (llegar hasta "TAREAS.C", insertar una tarea nueva y leer las tareas introducidas).
3. Finalmente, se comprobó la posibilidad de usar el mecanismo de control de procesos de Java (mediante la clase "Process") para gestionar un subproceso "s3270", el cual se encargará de realizar toda la comunicación con el sistema legado.

2. Estructura del wrapper con el sistema legado

Una vez se terminó el trabajo de exploración, se comenzó la implementación de la capa 3 (capas explicadas en el punto 3). El objetivo de ésta era ofrecer un interfaz de dos funciones a las capas superiores: `**newTask**`, la cual introducía una nueva tarea en el sistema; y `**getTasks**`, la cual leía las tareas almacenadas.

`**newTask**` es relativamente sencilla, dado que para su implementación únicamente se tuvo que tomar el diagrama de los pasos y programarlos. El único caso extraño era en el cual, durante la inserción, se requería avanzar la pantalla; para estos casos se creó la función `doubleEnter`, la cual se encarga de comprobar esos casos extremos y tratarlos adecuadamente.

`**getTasks**` es algo más compleja, dado que aparte de introducir toda la secuencia de comandos también tiene que interpretar la salida, convirtiéndola en un formato aceptable para las capas superiores del sistema. Dado que la información ya se nos da, dicho interprete no es especialmente complejo, siendo un sencillo iterador sobres las líneas de la pantalla, buscando los distintos patrones importantes (Fin de pantalla, tarea, final de lista de tareas...) y actuando en consecuencia. Al igual que en el caso previo, existe el caso en el cual la lista impresa puede cruzar el borde de la pantalla. En caso de que eso ocurra, se avanza la pantalla y se comienza de nuevo el proceso, hasta que ya no quedan tareas por leer. Este parser no toma en cuenta tareas multilínea (aquellas en las cuales la descripción se extiende más allá de una línea). Se han decidido eliminar en capas superiores, dado que la forma de almacenar datos del programa TAREAS.C daba problemas en esos casos, aun cuando el programa original permitía introducir ese tipo de datos problemáticos.

3. Análisis y diseño del sistema

Puesto que el problema al que hay que enfrentarse es un sistema legado que ha de ser dotado de una GUI que actúe como wrapper del mismo, se ha decidido crear un diagrama de capas del sistema para poder tener una mejor visión del mismo.

Ya que se dispone de la libertad de utilizar para la interfaz gráfica cualquier lenguaje y plataforma para acceder al mainframe, se ha decidido utilizar un sistema web dado que los miembros del equipo ya poseen experiencia desarrollando ese tipo de arquitecturas.

Para la correcta encapsulación del código y al tratarse de un wrapper a un sistema legado se ha tomado la decisión de crear una arquitectura de sistema de cuatro capas. La capa 1 encapsula la interfaz gráfica del cliente junto con el control de error en los "input" del usuario, asumiendo un papel de vista y de controlador gracias a la tecnología usada de AngularJS. La capa 2 asume el papel de controlador encapsulando el servidor web comunicando el cliente con la capa de acceso al mainframe. La capa 3 asume el papel del modelo de datos, y encapsula el acceso y la comunicación con la aplicación legada "TAREAS.C" en el sistema legado MUSIC/SP, ofreciendo para ello una interfaz java moderna. La capa 4 es el programa s3270 que controla la comunicación con el mainframe, y ofrece una interfaz por línea de comandos al mismo.

Las tecnologías usadas para las tres primeras capas han sido las siguientes:

- **Capa 1:** HTML5, CSS3, javascript, AngularJS y Bootstrap.
- **Capa 2:** Java Servlets.
- **Capa 3:** Java.

A continuación, se muestra el diagrama de capas y vista de módulos de la arquitectura del sistema:

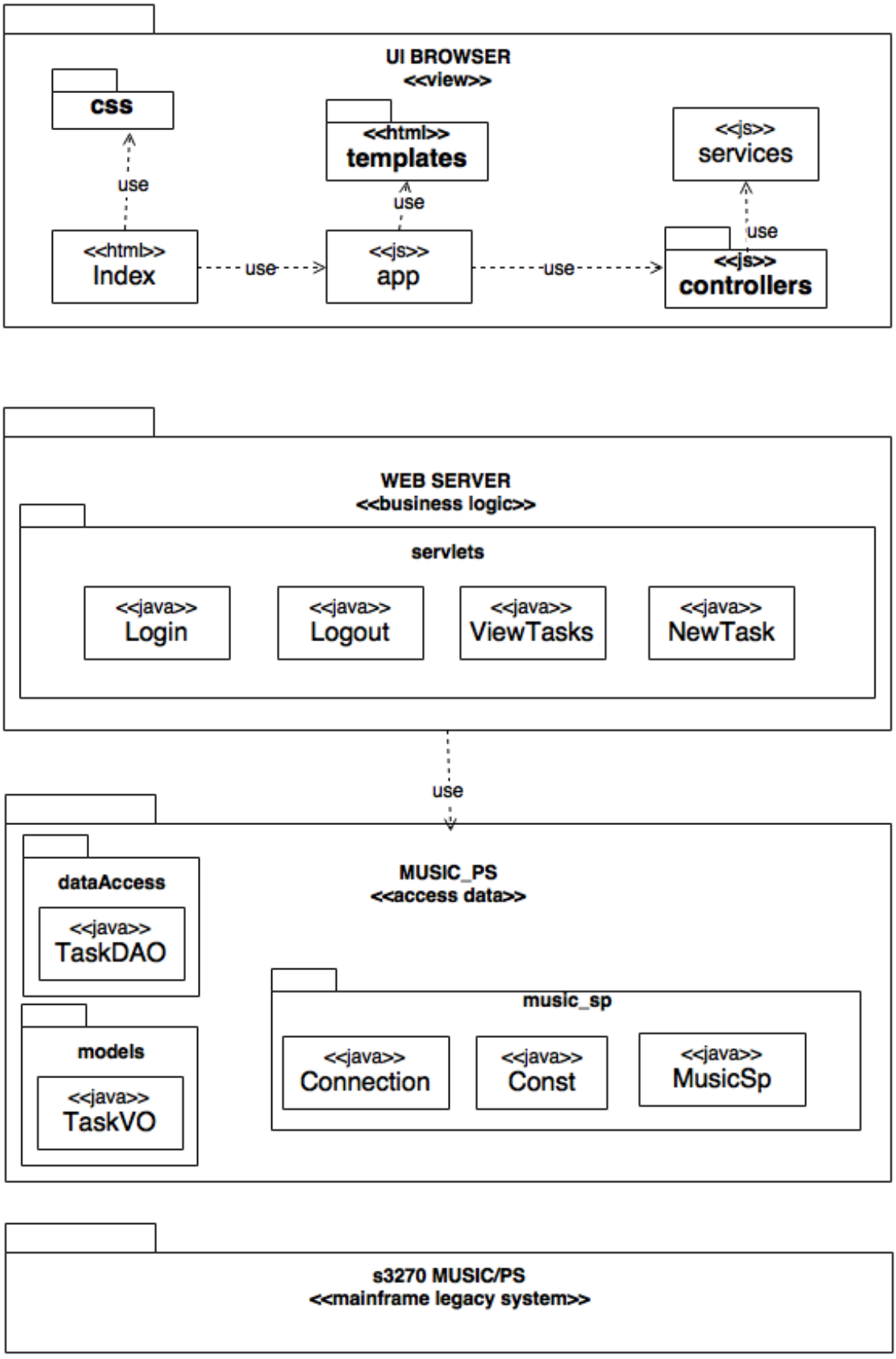

Figura 1. Diagrama de capas y vista de módulos de la arquitectura del sistema.

4. API del sistema

Para la comunicación entre cliente y servidor web se ha creado la siguiente API que recoge las 3 funcionalidades básicas que tiene que tener el sistema.

4.1. Login

GET /login

Request:

{
“user”:””
“pass”:””
}

Response:

Status 200

Status 404:

A String with a description message.

Nota: Los campos de la petición “user” y “pass” tienen que tener de 1 a 79 caracteres de longitud, y no contener espacios.

4.2. New Task

POST /newTask

Request:

{
“date”:””
“description”:””
“assignee”:””
}

Response:

Status 200

Status 404

A String with a description message.

Nota: La suma del campo de la petición “description” con el campo de la petición “assignee” no puede ser mayor de 12 caracteres en las tareas generales, y de 16 en las específicas, además de no contener espacios en blanco. El campo “date” debe seguir un formato DDMM y no contener espacios en blanco. El campo “assignee” debe ser vacío en caso de ser una tarea general.

4.3. View Tasks

GET /viewTasks

Request:

{
“type”:””
}

Response:

Status 200: Status 404

{
“tasks”: [
{“date”:””, ”description”:””, “assignee”:””},
{“date”:””, ”description”:””, “assignee”:””}
…
]
}

Nota: El campo de la petición “type” debe ser o “general” o “specific” dependiendo de si el listado de tareas que se quiere recibir es de tipo general o especifica respectivamente.

4.4. Logout

GET /logout

Request

Response:

Status 200 Status 404

5. Problemas encontrados y organización del trabajo

Los principales problemas a los que se ha enfrentado el grupo a la hora de realizar la práctica han sido los siguientes:

- La conexión con el sistema legado no es persistente, y cualquier información guardada en ella se borra cuando ésta se cierra. Esto originaba un problema, ya que se necesitaba guardar la conexión entre distintas peticiones del cliente (partiendo de la primera vez que hace una petición, que será el login). Para ello, el objeto de la conexión con el sistema legado se añade como atributo de sesión en la request del cliente en su primera petición. En las peticiones consiguientes, se cogerá la conexión de la request del cliente.
- Dado que el sistema legado sólo ofrecía una interfaz TUI, fue necesario crear un sencillo parser que se encargara de leer la información ofrecida por "s3270", y convertirla en un formato aceptable.
- Fue necesaria la sincronización entre nuestro sistema y el sistema legado, dado que si se envían varios comandos a gran velocidad, el sistema puede quedar en un estado inconsistente. Para ello, se utilizaron esperas pasivas. Existía la posibilidad

de realizar una espera activa contra el sistema legado, pero tenía el problema de sobrecargar el sistema, dado que no podía soportar el número de peticiones realizadas.

En lo referente al reparto de tareas y organización del trabajo, las partes de asegurar que la aplicación funciona correctamente, que se entiende el funcionamiento de la misma, el análisis y diseño, y la redacción de la memoria se ha realizado conjuntamente entre los 3 miembros del grupo de forma presencial o vía Skype participando y consensuando así las decisiones importantes de la práctica. Puesto que, como se ha comentado anteriormente, el equipo ya había trabajado con algunas tecnologías de las que se han usado y cada uno tenía un poco más de conocimiento con algunas de ellas, se ha dividido el trabajo de las tres capas para optimizar el proceso, encargándose Guillermo Robles de la capa 3, Iñigo Gascón de la capa 2, y Rubén Moreno de la capa 1.

6. Referencias

- Web de la asignatura *[http://webdiis.unizar.es/asignaturas/SL/]*
- s3270 Manual *[http://x3270.bgp.nu/Unix/s3270-man.html]*
- x3270 Manual *[http://x3270.bgp.nu/x3270-script.html]*
- x3270 Key Code Manual *[http://www.ibm.com/support/knowledgecenter/SSEQ5Y_6.0.0/com.ibm.pcomm.doc/books/html/emulator_reference11.htm]*