Segundo Obligatorio

Programación de Redes

Grupo N6A

2019

Docente:
Roberto Martín Assandri

Integrantes:
Rodrigo Bertolotti 220818
Rodrigo Duarte 217374

Introducción **3**

Requerimientos funcionales: **3**

RF1: Ver (y filtrar) un log de todos los eventos 3

RF 2: Dar de alta docentes 3

RF 3: Corrección de material por parte de docentes. 3

Requerimientos No Funcionales **3**

RF1: Actualizar el uso de threads a task 3

RF 2: Acceso a nuevas funcionalidades mediante HTTP. 3

RF3: Las nuevas funcionalidades en el viejo servidor no podrán ser expuestas mediante HTTP 3

Estructura del proyecto **4**

Diagrama de paquetes: 4

Diagrama de despliegue 4

RemoteService 5

CourseAPI 5

LogsAPI 5

Task **6**

MSMQ **6**

Remoting **6**

Web Services **6**

Funcionalidades no implementadas: **7**

Introducción

El objetivo del proyecto es simular una plataforma similar a la utilizada en la facultad, donde haya cursos disponibles para que los alumnos se inscriban, se puedan subir materiales y los profesores puedan corregirlos, notificando a los alumnos con su calificación.

El objetivo del trabajo es continuar desarrollando el sistema de la primer entrega, agregando nuevas funcionalidades mediante el uso de nuevas tecnologías. Las nuevas tecnologías a utilizar son:

- Manejo de Tasks
- Manejo de tecnologías MOM.
- Manejo de tecnologías RPC/ORB.
- Manejo de Web Services.

Requerimientos funcionales:

RF1: Ver (y filtrar) un log de todos los eventos

RF 2: Dar de alta docentes

RF 3: Corrección de material por parte de docentes.

Requerimientos No Funcionales

RF1: Actualizar el uso de threads a task

RF 2: Acceso a nuevas funcionalidades mediante HTTP.

RF3: Las nuevas funcionalidades en el viejo servidor no podrán ser expuestas mediante HTTP

Estructura del proyecto

Diagrama de paquetes:

Dividimos el proyecto en 9 paquetes diferentes, cada uno tiene una responsabilidad definida y sus funcionalidades son implementadas en las clases que son detalladas más adelante.

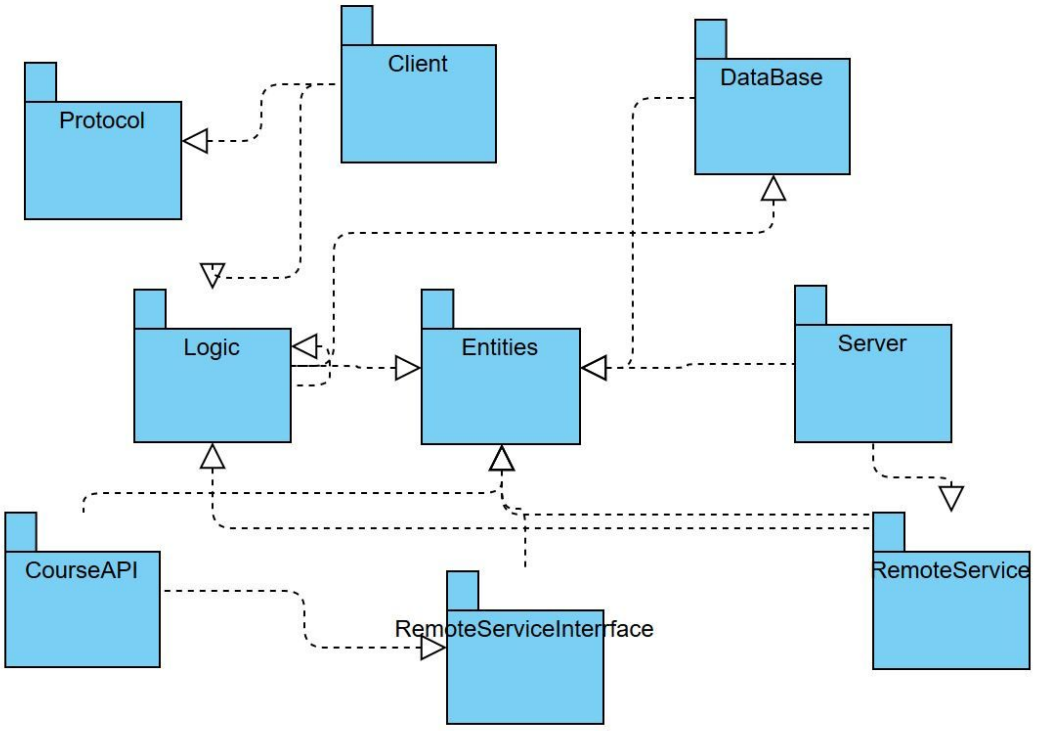
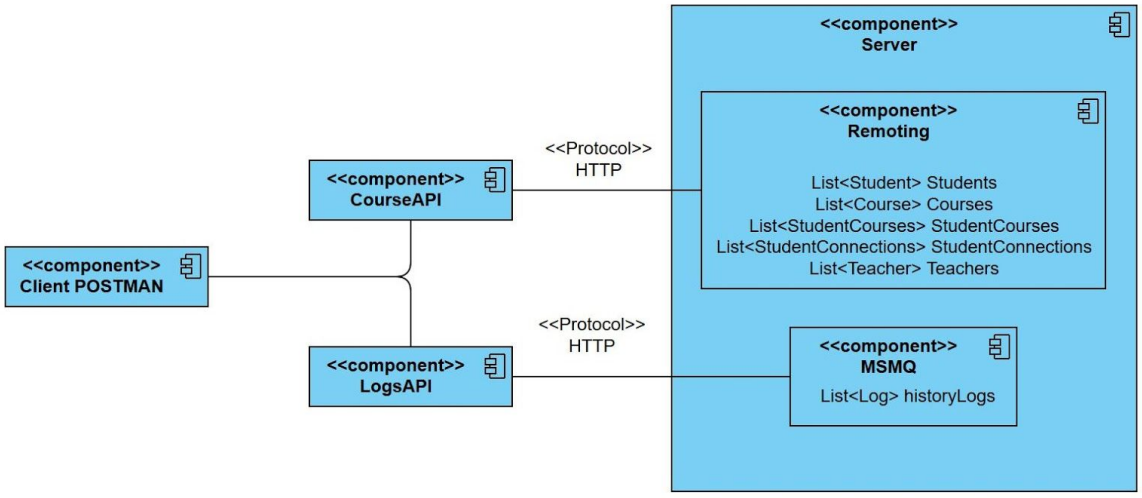


Diagrama de despliegue



Modificamos el viejo servidor para que sea servidor de remoting y de MSMQ. Evaluamos la opción de crear un servidor aparte para MSMQ pero no lo creímos necesario, por lo que lo incluimos dentro del viejo servidor. La API utilizada para las funcionalidades de docente consulta a el servidor de remoting, el cual contiene todas las listas de nuestra vieja clase Information y la de los logs consulta a la lista de logs (historyLogs), la cual se carga cada vez que se hace una llamada a la API, agregando el contenido de la cola a la lista y limpiandola.

Por lo tanto, tenemos un nuevo servidor que se encarga de manejar las listas que utiliza el resto del sistema, las cuales son llamadas por medio de RPC y por otra parte, también se encarga del funcionamiento de la queue de logs.

RemoteService

Servidor remoto, encargado de nuevas funcionalidades a implementar. Reemplaza a la clase Information de la entrega pasada ya que posee todas las listas. Estas listas las utilizan el resto de los métodos de la lógica del sistema y son accedidos mediante remote procedure calls. Se crea un objeto singleton para crear un acceso único a los datos de esta clase y que sea accedida la misma instancia desde el resto del sistema.

CourseAPI

API encargada de exponer las nuevas funcionalidades para los docentes.
Define 5 endpoints detallados más adelante para cumplir con los requerimientos.

LogsAPI

API encargada de exponer las nuevas funcionalidades para los logs. Se permite realizar llamadas a la api filtrando por su tipo.

Task

No implementado.

MSMQ

Utilizamos esta tecnología para registrar un historial de los eventos realizados en el sistema. Los eventos que se registran en el log son: alta de alumnos, alta de docentes, alta de cursos, baja de cursos, inscripciones a cursos y corrección de materiales.

Se crea una queue de nombre logqueue en la cual se guardan los eventos, los cuales tienen un tipo y una descripción. Cada vez que se realiza una de estas acciones se ingresa la acción a la cola. En el servidor de MSMQ se mantiene una lista llamada historyLog en la cual cada vez que se realiza una lectura a la cola, esta se vacía y se agregan los Logs (creados como una nueva entidad) a la lista historyLog, la cual es almacenada en memoria y persiste todos los logs del sistema. Luego mediante un cliente HTTP (en nuestro caso POSTMAN) podemos hacer llamadas a la API y filtrar las acciones por evento.

Para trabajar con logs agregamos una entidad Log al proyecto entities.

Remoting

Debido a que uno de los requerimientos no funcionales del sistema es que no se puedan exponer las nuevas funcionalidades pedidas por medio de HTTP, necesitamos utilizar un pasamanos entre el cliente y el servidor.

Nos encontramos con el siguiente problema. La “base de datos” debía ser la misma para las viejas y las nuevas funcionalidades, pero el nuevo servidor de Remoting que debía contener las nuevas funcionalidades no podía acceder a las listas ubicadas en el servidor anterior de forma local. Por este motivo decidimos transferir las listas de la clase Information a el

servidor remoto, y crear una instancia Singleton de dichas listas para el viejo servidor acceda a un objeto remoto de forma local (remoting) y no viceversa.

De esta manera creamos un servidor remoto que es accedido por el viejo servidor, el cual accede a sus funcionalidades de forma local.

Web Services

El acceso a las nuevas funcionalidades pedidas se debe realizar mediante HTTP, en nuestro caso utilizamos REST y POSTMAN como cliente para realizar funcionalidades pedidas.

Creamos dos proyectos para las APIs, CourseAPI y LogAPI, utilizadas para las nuevas funcionalidades de docentes y el filtro de logs respectivamente. Creamos 11 endpoints en total para cumplir con todas las funcionalidades.

POST api/Teachers/login: Login de docentes

GET api/Teachers/files: Devuelve la lista de materiales que un docente puede corregir.

PUT api/Teachers/files: Ingresa nota a un material por parte de un docente

GET api/Teachers: Devuelve la lista de docentes

POST api/Teachers: Crea docente

GET api/Logs: Devuelve la lista de logs filtrados por tipo

Funcionalidades no implementadas:

No se implementaron tasks.

Debido al pasamanos entre el servidor de la primera entrega y el remoto no pudimos mantener la lista de sockets por estudiantes conectados en el nuevo servidor. Por lo tanto, las nueva funcionalidad de asignar notas funciona correctamente, pero no se logra notificar al alumno, ya que no tenemos el socket que se encarga de las notificaciones.