RiveScript

Paradigmas de Programación (EIF-400) Spec: Rivescript SWI-Prolog

DR. CARLOS LORÍA-SÁENZ

LIBERADO: 30 DE AGOSTO 2019

EIF400 /UNA

Introducción

- Este documento (o SPEC) especifica los alcances y entregables del proyecto programado sobre Rivescript en SWI-Prolog del curso EIF400-II-2019-UNA
- Define los requerimientos funcionales mínimos esperados en distintos aspectos y facetas, incluyendo el stack de desarrollo obligatorio
- Define los mecanismos de organización, entrega, revisión y evaluación
- Incluye referencias a fuentes relevantes para realizar el proyecto

Objetivos Generales

- Poner en práctica técnicas de programación LP-FP-OOP aplicada en un caso de uso de programación Web y en un problema computacional no tradicional
- Experimentar con herramientas, paradigmas y tendencias actuales asociadas con el curso y alcances del proyecto
- Fomentar la investigación en temáticas relacionadas al curso de paradigmas según los requerimientos del proyecto lo demanden
- Documentar el proceso de investigación de manera formal
- Motivar al estudiante a ampliar su visión sobre temas alternativos en desarrollo de aplicaciones

Resumen de productos

- Una aplicación Web funcional y demostrativa de un motor (engine) original para procesar (un subset de) el DSL Rivescript en Prolog. El engine incluye cierto nivel de análisis estático antes de ejecutar código
- Casos de prueba (brains) demostrativos y originales del funcionamiento y alcances logrados
- Un reporte impreso de investigación sobre herramientas y enfoques investigados y usados para cumplir con los requerimientos de la aplicación

Requerimientos de organización

- Debe realizarse en los grupos previamente inscritos según se pidió al inicio del semestre
- No se acepta de otra forma
- Debe quedar constancia de las interacciones y contribuciones de los miembros en la herramienta colaborativa que usen

La Tarea

- Desarrollar una aplicación (app) original cliente-servidor que usando las herramientas y enfoques que se le piden en esta especificación le permita a sus usuarios los siguientes casos de uso:
  - Para un usuario de chat interactuar en una sesión cliente con bots escritos en el DSL Rivescript (RS) procesados estos en el server con un motor propio, teniendo la posibilidad de persistir sus sesiones en el sitio
  - Para un usuario admin, crear brains válidos en RS y persistirlos en el sitio de la app

Consideraciones iniciales

- La tarea requiere nociones básicas de programación Web en SWI-Prolog las que el profesor facilitará por medio de demos, en clase o en consulta extra y referencias a tutoriales
- Se explicará lo necesario para arrancar asumiendo trabajo colaborativo en el equipo
- Se entregan demos y referencias útiles que sirvan de modelo para realizar las tareas pedidas
- Se estima que con unas 4-6 horas de explicación se puede trabajar en el proyecto

Stack de Desarrollo: HTML+JS+CSS+PROLOG

- En el cliente
  - Arquitectura "single-page-application" (SPA)
  - HTML5, CSS3 y JS. Puede usar algún framework o biblioteca de cliente de su escogencia (que no sea un motor de Rivescript)
- En el server: SWI-Prolog usando websockets
- DCG para el Parser
- Paradigma: Patrones y estilos declarativos lógico-funcionales en ambas capas (cliente/server) donde corresponda incluyendo asincronía. ES>=6

Funcionalidades mínimas esperadas

- Una página de interacción para sesiones con un bot con historial persistente en el server por cada usuario
- Una página para administración de los brains (CRUD de brains)
- Un server que atienda ambas páginas separando presentación de servicios (chat y admin)
- Un servicio de login para autenticación en ambos casos de uso
- Una capa de persistencia en el server de sesiones de chat y de brains por usuario

Grados de libertad: presentación y persistencia

- Las páginas puede ser diseñadas de manera libre y original. Siguen un estilo común, tienen un about que indica el origen del proyecto ,autores y créditos
- Para el manejo de estilo y presentación se puede usar cualquier librería y herramienta que se pueda integrar bien con el mecanismo de servicio implementado
- Para la persistencia se puede usar en el file system o con un motor RDBMS (ambas son factibles en SWI-Prolog). La opción del RDBMS es libre en ese caso

Requerimientos desarrollo e implementación

- Eliminar lo más que se pueda todo patrón imperativo y estado mutable del código usando FP en el cliente y FP-LP el server donde se pueda.
- Organizar el código en módulos
- Usar arquitecturas SPA en cliente
- En el server, capa de servicios separada/desacoplable de la presentación en HTML
- Desarrollar un API que independice la capa de servicios de bot y su administración de la de presentación. Los servicios podrán ser usados sin la página, si fuera el caso, accediendo al API por HTTP

Requerimientos investigación

- Una parte del trabajo demanda investigar sobre temas y herramientas para cubrir los requerimientos:
- Web-framework en Prolog (contenido estático y dinámico)
- Persistencia en Prolog
- Unit testing en Prolog y JS
- Generación automatizada de documentación en Swi-Prolog y JS.
- Detalles del Motor de Rivescript (sintaxis y semántica del DSL)

Requerimientos Testing

- Desarrollar casos de prueba de brains que verifiquen que el motor trabaja correctamente
- Los casos de prueba son parte del proyecto y el entregable, reciben nota.
- Considere usar unit testing en Prolog

Requerimientos documentación

- Documentar de forma que se pueda generar automáticamente la misma
- La documentación recibe nota
- Considere usar pldoc o análogo

Evaluación y Valor App

- Demo en clase: 65% (unos 15 minutos por grupo)
- Mejor traerlo en su propia laptop por si acaso haya problemas de setup en el lab. Si no logran la demo por razones ajenas al profesor reciben cero
- Revisión de código 35%. Se revisará que se emplean técnicas FP-OOP, arquitectura y herramientas según lo solicitado
- Este proyecto se considera el primero de dos subproyectos sobre la misma temática, que así en conjunto cumplen con el rubro completo de Proyectos de la carta al estudiante
- Se puede otorgar hasta un 20% extra sobre la nota si se añaden más funcionalidades o herramientas (previamente discutidas con el profesor). Sólo aplica si se tiene un mínimo obligatorio (ver slide adelante)

Evaluación reporte temas y herramientas

- Este trabajo contribuye con el reporte en un porcentaje de 2.5% al rubro de investigación de la carta al estudiante, que evalúa la investigación y uso de herramientas que ayuden en la consecución del trabajo como las citadas en este SPEC.
- Se materializa en un reporte escrito a entregar sobre el desarrollo del proyecto y los resultados obtenidos (ver slide más adelante)

Uso, SCM, IDE, entrega

- Se espera que la aplicación pueda ser corrida usando desde una consola
- Pueden para su desarrollo usar cualquier IDE de su preferencia
- Pero: En todo caso, la construcción y corrida no deben depender de un IDE, durante la demo o al revisar offline el proyecto
- Debe usar un repositorio de manejo de fuentes versionados (SCM) como Github o comparable. Tenga en cuenta preservar la privacidad de su proyecto durante el curso.

Entregables: el código

- Se subirá según lo pedido al sitio dedicado para eso por el coordinador del grupo.
- El proyecto en un `.zip` (o equivalente) llamado `RS_Prolog_AAA_GG`, donde `AAA` es el nombre del coordinador y `GG-HH` es el grupo autor.
- Dentro del `zip` debe venir un directorio (carpeta) `rivescript` (con el proyecto bien estructurado) y un `README.txt` con los nombres de los autores, su horario y aspectos relevantes al revisor.
- Debe documentarse el proceso de corrida
- Incumplimiento de estos requisitos anula la revisión que hará el profesor offline (pierden así su 35%). Sin derecho a reclamo.

Entregables: el reporte

- Un reporte escrito(en papel) que documente el proyecto con estructura formal:
  - Portada
  - Introducción
  - Problema y su solución técnica
  - Análisis de Resultados
  - Instrucciones de uso
  - Limitaciones conocidas
  - Conclusiones
  - Referencias usadas
  - Entre 15 y 20 páginas Arial 12 pts espacio sencillo

Fecha y hora de entrega

- El día exacto de entrega se dará a conocer oportunamente en clase y por los medios usuales en el curso: esperado semana 10 (revisable)
- Ese día de entrega (mismo de la demo) se sube el entregable y se revisa la demo en la hora de matrícula. Esta dependa de la constitución del grupo: normalmente la hora de matrícula de la mayoría de miembros, salvo acuerdo distinto con el profesor
- En la demo se usa la misma versión entregada al profesor. Cualquier disparidad anula el proyecto. La prueba no debe depender de que haya Internet disponible.
- Entregas impuntuales por razones injustificadas podrán generar pérdida de puntos de hasta 10 puntos por minuto de atraso hasta llegar a cero.
- Es obligación estar presentes durante la demo. El ausente recibe cero completo en el trabajo.

Guía de revisión

- Se entregará oportunamente un documento con una guía de revisión la cual deberá ser impresa y traída el día de la demo. En ella se documentará las observaciones y nota por parte del profesor.
- Se solicitarán casos de prueba mínimos (brains) que se deberán probar con la aplicación
- Podrán haber casos de prueba sorpresa conocidos sólo el día de la demo.

La Demo en Clase

- Presentan la guía de revisión y el reporte escrito
- Se demuestran las funcionalidades pedidas de manera expedita y eficiente en el margen de tiempo asignado a cada grupo.
- Cualquier miembro debe estar en capacidad de contestar preguntas técnicas sobre aspectos del proyecto
- EL profesor podrá seleccionar a su criterio a la persona a contestar (los demás no participan)
- La calificación de esas respuestas puede ser de alcance individual o grupal a criterio del profesor

Extras

- Se podrán desarrollar extras que pueden aportar hasta un 20% sobre lo nota obtenida, siempre que esta nota alcance al menos un 90% de lo considerado como obligatorio
- Los grupos pueden proponer extras, los que pueden incluir frameworks de cliente, funcionalidades adicionales a las interacciones (audio, imágenes…), extensiones originales a Rivescript, etc. No todos los extras valen igual.
- Los posibles extras deberán ser primero aprobados formalmente por el profesor, **antes** de su incorporación al proyecto. De lo contrario serán ignorados.

Referencias Externas

- Ogborn, A Tutorial Creating Web Applications with SWI-Prolog. Github de los ejemplos
- Ogborn, A. Using DCG in SWI-Prolog. Github de ejemplos.
- Petherbridge, N. Sitio Rivescript. Ver tutorial y Working Draft SPEC.
- Petherbridge, N. Overdijk, M. Github de A RiveScript interpreter for Java.
- Wielenmaker, J. SWI-Prolog HTTP Support.

Demo: Se adjunta demo RS-Prolog

Será explicado en clase