Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de Computación
IIC2143 – Ingeniería de Software (I/2018)

Proyecto semestral

1. Objetivos

- Aplicar metodologías ágiles en el contexto de un equipo de desarrollo.
- Aprender el *framework Ruby on Rails* para desarrollar aplicaciones web.
- Aprender a usar distintas herramientas por cuenta propia y explorar distintas soluciones dependiendo de las exigencias del cliente o *product owner*.
- Conocer sobre buenas prácticas y herramientas de desarrollo de *software*.

2. Introducción

Hoy en día el internet y la web son el principal punto de encuentro entre personas ya que proveen gran facilidad de comunicación. Es común hallar en muchos sitios largas discusiones sobre cualquier tipo de tema posible: política, fútbol, cocina, programas de televisión e incluso poesía. Sin embargo, estas conversaciones están repartidas en distintas páginas, no se encuentran organizadas por dichos temas y se pierden fácilmente entre tantas publicaciones aisladas. Por esto, es difícil encontrar discusiones concentradas a algún tema de interés en la gran vastedad del internet. Se vuelve necesaria la existencia de una plataforma donde sea posible encontrar fácilmente **comunidades** sobre **cualquier tema** imaginable. De esta forma se podrán encontrar y crear rápidamente conversaciones con personas interesadas en temas similares. A su equipo, como desarrolladores de *software*, se les encomienda crear una aplicación web que solucione dicha necesidad: una plataforma de foros de discusión.

3. Características generales

La aplicación a desarrollar debe permitir a los usuarios acceder a distintos foros de discusión, ver publicaciones, comentar y administrar sus publicaciones o foros favoritos. Cada foro tiene un tema específico del cual tratan sus publicaciones. Luego, cada usuario es capaz de subscribirse a todos los foros que sean de su interés. Así, puede leer todas las discusiones que se forman en cada foro. Una publicación de un foro es creada por algún usuario para comenzar una nueva discusión. Otros usuarios pueden comentar ésta, para así formar conversaciones dentro de la publicación. Además, los usuarios pueden votar a favor o en contra de una publicación o comentario, dependiendo de si es de su gusto. En consecuencia, las publicaciones y comentarios dentro de estas ganan popularidad dependiendo del número de votos a su favor. A su vez, los usuarios autores de dichas publicaciones y comentarios ganan cierta

reputación que aumenta con el total de votos a favor que recibe como autor. De cada foro están a cargo usuarios moderadores, capaces de administrar las publicaciones creadas en el foro y configurarlo estéticamente.

Por ejemplo, el usuario `rasaffie` disfruta aprender sobre café y sus distintas cepas. Decide revisar la plataforma en su foro de café y encuentra nuevas publicaciones sobre la apertura de una cafetería en Santiago centro que él visitó recientemente. Selecciona la primera y comenta sobre su experiencia en dicho local. Luego, al usuario se le ocurre buscar la existencia de un foro sobre trompos, ya que es su *hobby* coleccionarlos desde distintas procedencias. Busca y encuentra dicho foro. Navega y encuentra publicaciones con fotos de trompos que ha estado buscando hace un tiempo. Decide suscribirse a su nuevo foro favorito y guarda las publicaciones que le interesan, para futura lectura. Finalmente vuelve a la publicación del nuevo local de café, y se alegra al ver que su comentario recibió 23 votos a favor y varias preguntas. ¡Su reputación aumentó!

Como se explicó, la plataforma busca encapsular varias comunidades de intereses distintos bajo un mismo techo. A continuación, se listan aspectos básicos que la plataforma debe cubrir.

3.1. Comportamiento general

- La aplicación debe permitir el registro y autentificación de usuarios.
- Un usuario registrado puede modificar sus datos de cuenta.
- Un usuario registrado puede publicar, comentar, responder y votar a favor o en contra en cualquier foro.
- Una publicación o comentario obtiene un puntaje de reputación determinado por la diferencia entre votos a favor y votos en contra.
- Un usuario obtiene un valor total de reputación determinado por la suma de puntajes de reputación de sus publicaciones y comentarios.
- Un usuario registrado puede suscribirse y desuscribirse de cualquier foro.
- Un usuario registrado puede guardar publicaciones favoritas.

3.2. Tipos de usuario

Su aplicación debe manejar los siguientes tipos de usuarios:

- **Común**: Es el usuario común descrito hasta el momento, capaz de navegar, publicar, comentar y votar. Este usuario está registrado y mediante autentificación es capaz de acceder a estas acciones.
- **Visita**: Usuario no registrado en la plataforma. Es sólo capaz de navegar en foros y leer publicaciones. No puede publicar, comentar, votar, o interactuar con usuarios registrados a través de la plataforma.
- **Moderador**: Es un usuario común registrado en la plataforma, pero con mayores facultades. Además de las características de usuario común, puede eliminar publicaciones y comentarios de otros usuarios dentro de un foro con el fin de mantener su bienestar. Para obtener estas facultades, el usuario debe solicitar convertirse en moderador. En la siguiente sección se especifica en detalle este comportamiento.
- **Administrador**: Usuarios que administran la plataforma, con la capacidad de moderar, crear, editar y eliminar foros completos.

3.3. Moderación

Cada foro puede tener usuarios moderadores. Éste es un título que se le puede otorgar a un usuario común dentro de un foro. Solo al alcanzar cierta reputación en la plataforma, un usuario puede solicitar ser moderador de un foro específico. Esta solicitud puede ser solo aceptada o rechazada por otro moderador de dicho foro (o administrador del sistema). Una vez aceptado como moderador, el usuario es capaz de eliminar publicaciones y comentarios que no sigan las reglas del sitio, pero solo en el foro donde fue nombrado moderador. Además, es capaz de publicar y comentar como usuario común.

3.4. Tipos de publicación

Su aplicación debe manejar los siguientes tipos de publicación:

- **Texto**: El tipo más básico de publicación, que solo incluye contenido en forma de texto.
- **Imagen**: Publicación cuyo contenido es una imagen.
- **Enlace**: Publicación cuyo contenido es un hipervínculo.
- **Encuesta**: Publicación que propone una pregunta para encuestar a sus lectores. Además de contener una pregunta, contiene opciones de respuesta por las cuales un usuario común puede votar.

4. Atributos mínimos

4.1. Usuario

Debe manejar al menos los siguientes aspectos de un usuario:

- Nombre.
- Correo electrónico.
- Reputación.
- Imagen de perfil.
- Suscripciones.
- Publicaciones favoritas.

4.2. Foro

Debe manejar al menos los siguientes aspectos de un foro:

- Nombre.
- Tema.
- Descripción.
- Número de suscriptores.

4.3. Publicación

Debe manejar al menos los siguientes aspectos de una publicación:

- Título.
- Autor.
- Fecha de creación.
- Foro al que pertenece.
- Tipo de publicación.
- Contenido.
- Puntaje de reputación.

4.4. Comentario

Debe manejar al menos los siguientes aspectos de un comentario de respuesta:

- Publicación o comentario al que responde.
- Autor.
- Contenido.
- Puntaje de reputación.
- Fecha de creación.

5. Estadísticas

Finalmente, se consideran los siguientes aspectos estadísticos para la plataforma:

- Usuarios pueden ver un *ranking* de foros con más suscriptores.
- Usuarios pueden ver un *ranking* de usuarios con mayor reputación.
- Administradores tienen acceso a un *dashboard* de estadísticas generales de la aplicación. Ejemplos de relaciones son:
  - Distribución de suscriptores en foros.
  - Estadísticas y distribuciones de actividad de usuarios: publicaciones, comentarios o votos.

6. Funcionalidades mínimas

Su aplicación debe abarcar las siguientes funcionalidades mínimas:

- CRUD[1] de usuarios.
- CRUD de foros.
- CRUD de publicaciones.

[1] ***C****reate,* ***R****ead,* ***U****pdate and* ***D****elete.*

- CRUD de comentarios.
- *Sign up* de usuario.
- *Log in* de usuario.
- Actualización de información de cuenta de usuario.
- Votación de publicaciones y comentarios.
- Cálculo de reputación de publicaciones, comentarios y usuarios.
- Búsqueda de foros.
- Exploración de foros populares o recientemente creados.
- Búsqueda de publicaciones dentro de un foro.
- Exploración de publicaciones populares o recientemente creadas.
- Administrar suscripciones de foros de usuario.
- Administrar publicaciones favoritas de usuario.
- Solicitud y respuesta de nombramiento de moderadores.

7. Referencias

Las siguientes plataformas son casos reales con características similares a las pedidas:

- Reddit
- Stack Overflow
- Quora

8. Requisitos mínimos de desarrollo

Para asegurar un producto de calidad, se les pide que utilicen las siguientes herramientas y buenas prácticas de desarrollo de *software*. Todas ellas son un estándar básico para la industria de software actual y potencian la producción de equipos de desarrollo.

8.1. *Kanban*: *Trello*

Utilizar el servicio de *Kanban* *Trello* para organizar su trabajo como equipo. Cada equipo debe tener un tablero de *Trello* que compartirá con su *product owner*. Éste puede tener la estructura (columnas) que el equipo encuentra conveniente mientras se note un claro flujo de trabajo que comunique el estado del proyecto a su *product owner*.

8.2. *Gitflow*

Para desarrollar la aplicación, gestionarán su proyecto en un repositorio *git*. Sobre esto, deben seguir el modelo de uso *Gitflow*. No es necesario seguirlo al pie de la letra, mientras se ocupen al menos dos *branches* principales y una *branch* por funcionalidad .

8.3. *Rubocop*

Seguir alguna guía de estilo de código para *Ruby* monitoreado por la gema *Rubocop*. Las configuraciones de estilo quedan a decisión de cada grupo, pero una vez fijadas deben respetarse.

8.4. *Heroku*

Utilizar la plataforma *Heroku* para publicar sus aplicaciones a producción.

8.5. *Docker*

Configurar sus ambientes de desarrollo con *Docker*.

9. Entregas, hitos y evaluación

El proyecto se llevará a cabo mediante desarrollo ágil inspirado en *Scrum*. Cada entrega se separa en un *Sprint* distinto, donde el trabajo para cada *Sprint* es negociado con su *product owner* en reuniones de *Sprint Review*.

9.1. *Product owner* y *Sprint Review*

Cada grupo de desarrollo tendrá asignado un *product owner* (ayudante) quien actúa como la contraparte del proyecto. Tras cada término de *Sprint* (entrega) se debe agendar una reunión (*Sprint Review*) con su *product owner* para discutir y monitorear el avance del proyecto. Además, deben definir junto a ella o él los pasos a seguir para el siguiente *Sprint*. **Todos los miembros del equipo deben asistir al *Sprint Review*** y debe planificarse para realizarse entre los **tres** inmediatamente siguientes días hábiles después del fin de un *Sprint*.

9.2. Coevaluación

Por cada entrega deberá responderse una coevaluación de sus compañeros. Ésta puede afectar positiva o negativamente su calificación. Detalles de ésta se especificarán luego de la primera entrega.

9.3. Evaluación

Su ayudante asignado es el encargado de evaluar su avance, además de llevar el rol de *product owner*. Para cada *Sprint Review*, su ayudante hará una sesión de corrección que dependerá de la entrega. Ésta puede implicar evaluación grupal y/o evaluación individual de conocimientos. Las notas parciales de cada entrega son **individuales** y consideran el avance grupal, individual y la coevaluación respondida.

9.4. Entregas

En total, son 5 entregas parciales. Cada entrega se realiza mediante su repositorio asignado de grupo en la organización de *GitHub* del curso, donde se corregirá el último *commit* en la rama *master* dentro de plazo. Luego de la entrega 0, todas incluyen avance de funcionalidades. Cuáles de ellas deben incluir en cada entrega depende de sus negociaciones con su *product owner*. Si bien este documento sirve como guía base de proyecto, **el resultado final**

puede (y debe) variar. Adicionalmente, algunas entregas incluyen un aspecto obligatorio o evaluación específica a realizar. A continuación, se listan a grandes rasgos los entregables:

9.4.1. Entrega 0 (29 de Marzo)

Relatos de usuario y aplicación mínima "Yay! You're on Rails!" publicada en *Heroku*.

9.4.2. Entrega 1 (21 de Abril)

Funcionalidades y modelación mediante diagrama E/R de la aplicación.

9.4.3. Entrega 2 (12 de Mayo)

Funcionalidades y evaluación individual de conocimientos *Ruby on Rails*.

9.4.4. Entrega 3 (2 de Junio)

Funcionalidades y segunda evaluación individual de conocimientos *Ruby on Rails* como oportunidad de corrección de nota.

9.4.5. Entrega 4 (23 de Junio)

Funcionalidades e integración con una *API* a elección.

9.5. Presentación final (TBA)

Finalmente, luego de las entregas parciales se realizará una presentación del producto logrado al equipo docente del curso. En esta oportunidad se busca que el equipo de desarrollo **completo** presente lo experimentado durante el desarrollo, el resultado obtenido y las lecciones aprendidas.

9.6. Nota

Como se especifica en el programa del curso, la nota de proyecto se divide en tres componentes:

- $\overline{E}_P$: Promedio de notas de entregas parciales.
- E_F: Nota de entrega final, como producto desarrollado.
- P_F: Nota de presentación final.

La nota de proyecto (P) se calcula como sigue:

$$P = 0{,}5 \cdot \overline{E}_P + 0{,}2 \cdot E_F + 0{,}3 \cdot P_F$$

10. Política de integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Academica publicado por la Dirección de Docencia de la Escuela de Ingeniería en el SIDING.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo"se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1.1 en el curso y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por çopia"se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente. Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especicadas en dicho reglamento.