Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación
IIC2143 - Ingeniería de Software (I/2019)

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

Día a día, la comunidad de estudiantes de la Pontificia Universidad Católica de Chile ha necesitado hacer un uso eficiente de salas de estudio, tanto para conocer el estado de estas salas, organizar grupos de estudio y/o clases particulares. A tu equipo de desarrollo le han pedido implementar una plataforma que le permita al alumnado UC: compartir información de las salas de estudio, estudio en grupo, clases particulares y apuntes para organizar una buena jornada de estudio.

3. Características Generales

La aplicación a desarrollar debe permitir a los usuarios acceder a las salas de estudio, y en cada sala de estudio poder ver los distintos grupos de estudio y clases particulares programadas en ella. En cada una de estas, el usuario puede ver el espacio disponible en la sala, la cantidad de personas dentro del grupo de estudio, el curso que se esta estudiando dentro del grupo. Por otra parte, la aplicación debe permitir además a los usuarios publicar grupos de estudio y apuntes sobre un curso y también que puedan comentar y evaluar las publicaciones de otros usuarios.

Por ejemplo: El usuario `jjaguillon` tiene planeado estudiar los contenidos del curso `IIC2143` y quiere saber si ya existe un grupo de usuarios estudiando en grupo. Para eso, ingresa a la aplicación, y busca en el campus San Joaquín si hay grupos de estudio sobre el curso en cuestión, al no encontrar ninguno decide publicar un grupo de estudio en la sala de estudios del Hall de Estudiantes. Luego, el usuario `jerosalazar` encuentra el evento y se une mediante la plataforma, llegando al lugar 5 minutos después de su confirmación. Al finalizar el el estudio, el usuario `jerosalazar` decide publicar un resumen que creo en su tiempo de estudio en la plataforma, ingresa a la página y busca la ventana respectiva al curso, realizando la publicación, la cual recibe 2 comentarios con dudas y correcciones en el resumen, más 50 votos a favor y 3 en contra.

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 comentar y evaluar las salas de estudio
- Un usuario registrado puede dar votos positivos o negativos y comentar las publicaciones de otras personas
- Un usuario registrado puede unirse a un evento dentro de la sala de estudio
- Un usuario registrado puede crear un evento dentro de la sala de estudio
- Un usuario registrado puede calificar la sala de estudio luego de un evento, con respecto a: disponibilidad, ruido, cantidad de enchufes.
- Un usuario registrado puede suscribirse y desuscribirse a cursos como alumno
- Un usuario registrado puede suscribirse y desuscribirse a cursos como profesor particular
- Las publicaciones están clasificadas por curso
- Un usuario registrado puede guardar publicaciones favoritas
- Un usuario registrado puede enviar mensajes a otro usuario registrado

3.2. Tipos de usuario

Su aplicación debe manejar los siguientes tipos de usuario:

- **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.
- **Moderador:** Es un usuario común registrado en la plataforma, pero con mayores facultades. Además de las características de un usuario común, puede eliminar publicaciones, eventos y comentarios de otros usuarios dentro de un curso con el fin de mantener su bienestar y no romper con algunas reglas del curso. 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 cursos, salas de estudio y campus completos.

3.3. Moderación

Cada curso puede tener usuarios moderadores. Éste cargo se le otorga a quien tenga relación con la administración del curso (por ejemplo: ser ayudante del ramo), por lo que debe enviar una solicitud con alguna imagen que demuestre tal cargo. Esta solicitud solo puede ser aceptada o rechazada por otro moderador del curso (o administración del sistema). Al ser aceptado, el usuario es capaz de eliminar publicaciones y comentarios que no sigan las reglas del curso, pero solo en el curso donde fue nombrado moderador. Además, es capaz de publicar y comentar como usuario común.

3.4. Tipos de evento

Su aplicación debe manejar los siguientes tipos de evento:

- **Grupo de estudio:** Este evento le informa a los usuarios lo siguiente:
  - Curso que se esta estudiando
  - Cantidad de usuarios que permite el grupo de estudio
  - Los usuarios actuales dentro del grupo
  - Hora de posible termino (Se deja a su criterio la duración del evento)
- **Clases particulares (Busco):** Este evento le informa a los usuarios lo siguiente:
  - Curso que se está buscando profesor particular
  - Los usuarios que quieren clases
  - Hora de posible termino (Nuevamente se deja a su criterio la duración del evento)
- **Clases particulares (Ofrezco):** Este evento informa a los usuarios:
  - Usuario que ofrece clases
  - Precio ofrecido por la clase
  - Hora de posible termino (Se deja a su criterio)
- **Sala ocupada:** Este evento informa a los usuarios el uso de esta sala para un evento externo (ejemplo: una charla), con su hora de termino respectiva (En este caso, de manera obligatoria).

4. Atributos mínimos

4.1. Usuario

Debe manejar al menos los siguientes aspectos de un usuario:

- Nombre
- Correo
- Imagen de perfil
- Cursos donde es alumno
- Cursos donde es profesor particular
- Publicaciones favoritas

4.2. Campus

Debe manejar al menos los siguientes aspectos de un campus:

- Nombre
- Ubicación
- Mapa

4.3. Sala de estudio

Se debe manejar al menos los siguientes aspectos de una sala de estudio:

- Nombre
- Ubicación
- Puntaje de disponibilidad
- Puntaje de ruido
- Puntaje de enchufes

4.4. Evento

Se debe manejar al menos los siguientes aspectos de un evento:

- Autor
- Fecha y hora de creación
- Sala de estudio perteneciente
- Tipo de evento

4.5. Curso

Debe manejar al menos los siguientes aspectos de un curso:

- Nombre
- Sigla

4.6. Publicación

Se debe manejar al menos los siguientes aspectos de un curso:

- Titulo
- Autor
- Fecha y hora de creación
- Curso al que pertenece
- Contenido
- Descripción
- Puntaje de reputación

4.7. Comentario

Se debe manejar al menos los siguientes aspectos de un comentario:

- Publicación al que responde
- Autor
- Fecha y hora de creación
- Contenido
- Puntaje de reputación

5. Información que se puede proporcionar

Pueden usar su creatividad pero aquí se ofrecen algunas ideas:

1. Usuarios pueden ver las últimas publicaciones de los cursos que está cursando
2. Usuarios pueden ver las publicaciones con mayor reputación de los cursos que esta dando
3. Usuarios pueden filtrar los resultados de los puntos 1. y 2.
4. Usuarios pueden ver los últimos eventos de los cursos que está cursando
5. Moderadores pueden ver las últimas publicaciones de los cursos que está moderando
6. Administradores tienen la distribución de usuarios en cursos
7. Administradores tienen las estadísticas y distribuciones de actividad de usuarios: publicaciones, comentarios y eventos

6. Funcionalidades mínimas

Su aplicación debe abarcar las siguientes funcionalidades mínimas:

- CRUD[1] de usuarios.
- CRUD de campus.
- CRUD de salas de estudio.
- CRUD de eventos.
- CRUD de cursos.
- CRUD de publicaciones.
- CRUD de comentarios
- *Sign up* de usuario.
- *Log in* de usuario.
- Actualización de información de cuenta de usuario.
- Votación de salas de estudio (Disponibilidad, Ruido, Enchufes).
- Búsqueda de eventos.
- Búsqueda de cursos.
- Votación de publicaciones y comentarios.
- Búsqueda de publicaciones dentro de un curso.
- Administrar suscripciones de cursos de usuario.
- Administrar publicaciones favoritas.
- Solicitud y respuesta de nombramiento de moderadores.

[1] *Create, Read, Update and Delete*

7. 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.

7.1. *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*.

7.2. *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.

7.3. *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.

7.4. *Heroku*

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

7.5. *Docker*

Configurar sus ambientes de desarrollo con *Docker*.

8. 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*.

8.1. *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*.

8.2. 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.

8.3. 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.

8.4. Entregas

En total, son 4 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:

8.4.1. Entrega 0 (7 de Abril)

Relatos de usuario y aplicación mínima "Hello World!" publicada en *Heroku*.

8.4.2. Entrega 1 (27 de Abril)

Funcionalidades, modelación mediante diagrama E/R de la aplicación, evaluación individual de conocimientos sobre *Docker*.

8.4.3. Entrega 2 (18 de Mayo)

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

8.4.4. Entrega 3 (11 de Junio)

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

8.5. 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.

8.6. 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:

$$P = 0.5 \cdot \overline{E}_P + 0.2 \cdot E_F + 0.3 \cdot P_F$$

9. Recomendaciones Finales

Tienen bastante libertad para construir una aplicación de acuerdo a lo que cada grupo considere que es importante dado el espíritu y los objetivos que se explicaron al comienzo. Las funcionalidades (features) deben ser negociadas con el *product owner* (el ayudante que les seguirá durante todo el semestre) Se recomienda identificar y levantar la mayor cantidad posible de épicas y relatos de usuario en la *Entrega o Sprint 0* a pesar que no terminen siendo todos finalmente implementados.

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 "copia" 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 especificadas en dicho reglamento.