Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de Computación
IIC2143 – Ingeniería de Software (II/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

La cantidad de gente que tiene la posibilidad de viajar es cada día mayor, y con esto ha aumentado la necesidad de obtener información de lugares turísticos, tener datos de hoteles y ver referencias de los distintos aspectos de un viaje (calidad hotel, vuelos, restaurantes, etc..) para aprovechar al máximo la experiencia de viajar. Como las agencias de viaje no siempre son una buena fuente de información, una empresa te ha pedido que los ayudes a entrar al mundo del internet para cubrir la creciente demanda de información por parte de la comunidad viajera. A tu equipo de desarrollo le han pedido que implementen una plataforma que permita a la comunidad viajera compartir información de restaurantes, hoteles, lugares para visitar y actividades que pueden hacer en las distintas ciudades, para poder planificar sus vacaciones con la mayor información posible y para poder evaluar y contar las propias experiencias de sus viajes.

3. Características generales

La aplicación a desarrollar debe permitir a los usuarios acceder a distintas ciudades, y en cada ciudad puede ver distintos hoteles, restaurantes y lugares turísticos de interés. En cada uno de estos, el usuario puede ver una evaluación promedio del lugar, comentarios de otras personas y además puede evaluar y dejar un comentario de dicho lugar. Para permitir dar más confiabilidad a los usuarios que den comentarios mas fiables, uno debe poder votar de forma positiva o negativa a los comentarios de otros usuarios, y debe poder ver la reputación de dicho usuario (Sistema de reputación se detalla más adelante).

Por ejemplo: El usuario `hmuller` tiene un viaje planeado a Rio de Janeiro y quiere obtener información para saber que paseos hacer, en qué hotel hospedarse y ver los posibles restaurantes del lugar. Para esto, ingresa a

la página, y entra a ver la información que hay de Rio de Janeiro. En la lista de hoteles, encuentra uno que le gustó porque tiene un buen promedio de evaluaciones. Para ver más detalles, ingresa a dicho hotel para ver los comentarios que han dejado los usuarios, y ve que el usuario `ifcarrera` ha dejado un comentario muy positivo, y dicho comentario tiene 100 votos a favor y el usuario tiene una alta reputación, por lo que decide creerle y se decide por ir a ese hotel. (Ojo: La plataforma no permite hacer reservas, solo permite obtener información). Después recuerda su viaje a Nueva York, y recuerda su buena experiencia en el restauran Benihana, por lo que decide dejar un comentario recomendando el lugar. Para esto, vuelve a la lista de lugares, y va a Nueva York. Luego encuentra el restaurant Benihana y deja un comentario para que el resto de la comunidad lo pueda ver.

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 lugares a los que ha ido.
- Un usuario registrado puede dar votos positivos o negativos a los comentarios de otras personas.
- 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 usuarios capaces de publicar para seguir sus actividades.
- Los lugares están clasificados por países.
- 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, 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 las ciudades y leer las distintas cosas que tiene cada ciudad. 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 país puede tener usuarios moderadores. Éste es un título que se le puede otorgar a un usuario común dentro de un país. Solo al alcanzar cierta reputación en la plataforma, un usuario puede solicitar ser moderador de un país específico. Esta solicitud puede ser solo aceptada o rechazada por otro moderador de dicho país (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 país 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.
- **Ubicación**: publicación la cual incluya ubicación en el mapa.

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.
- Lugares favoritos.

4.2. País

Debe manejar al menos los siguientes aspectos de un país:

- Nombre.
- Descripción.
- Número de suscriptores.
- Ubicación.

4.3. Ciudad

Debe manejar al menos los siguientes aspectos de una ciudad:

- Nombre
- Descripción
- Ubicación

4.4. Itinerarios

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

- Nombre
- Descripción
- Lugares o actividades a realizar
- Duración

4.5. Publicación

Debe manejar al menos los siguientes aspectos de una publicación:

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

4.6. 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 ciudades/países 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 países/ciudades.
  - 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 países.
- CRUD de ciudades.
- CRUD de itinerarios.
- CRUD de publicaciones.
- 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 países y ciudades.
- Exploración de lugares populares o recientemente creados.
- Búsqueda de publicaciones dentro de una ciudad.
- Exploración de publicaciones populares o recientemente creadas.
- Administrar suscripciones de países/ciudades de usuario.
- Administrar publicaciones favoritas de usuario.
- Solicitud y respuesta de nombramiento de moderadores.

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

7. Referencias

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

- TripAdvisor
- Triposo
- FourSquare

*Ojo: No se les pide que imiten estás páginas, sino que las tengan como referencia para ideas.

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*. Además, antes de cada *Sprint Review* se debe subir en el repositorio asignado al grupo la propuesta de avance para el próximo *Sprint* en una carpeta llamada propuesta. De no entregar este documento, se aplicará un descuento en el *Sprint Review*.

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. Propuesta

Para cada entrega, exceptuando la última, el grupo deberá subir al repositorio una propuesta de features que desea realizar para el próximo *Sprint*. Este será evaluado por el *Product Owner*, él cual puede realizar modificaciones como estime conveniente. El no entregar esta propuesta antes del *Sprint Review*, el grupo será penalizado en la evaluación del *Sprint* que se está evaluando en ese momento.

9.4. 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.

9.5. 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:

9.5.1. Entrega 0 (28 de Agosto)

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

9.5.2. Entrega 1 (14 Septiembre)

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

9.5.3. Entrega 2 (13 de Octubre)

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

9.5.4. Entrega 3 (3 de Noviembre)

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

9.5.5. Entrega 4 (24 de Noviembre)

Funcionalidades e integración con una *API* a elección.

9.6. 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.

9.7. 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 "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.