



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2143 - INGENIERÍA DE SOFTWARE 2022-1

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

Dado que volvimos a clases a presenciales, los alumnos del DCC están buscando maneras seguras y ecológicas para ir a la universidad. Para esto les han pedido desarrollar una aplicación web que le permita a los estudiantes hacer turnos para trasladarse en auto durante la semana y así evitar exponerse lo más posible en otros medios de transporte como metro o micro, además de cuidar el medio ambiente.

3. Características Generales

La aplicación debe permitir a los visitantes ver información sobre el sitio y su funcionamiento. Una vez que un visitante se registra e inicia sesión, éste podrá subir una foto e información personal. Luego tendrá la opción de subir una publicación para ofrecer un turno de ida o vuelta, indicando el día y la hora en que salen. En el caso de que un usuario encuentre una publicación que le interese, este podrá mandar una solicitud para ser parte del turno. Por otro lado, el dueño de la publicación recibirá esta solicitud y tendrá la opción de aceptarla o rechazarla. En el caso de aceptarla, se habilitará una opción para que ambos usuarios, además del resto de las personas que forman parte del turno, puedan conversar, entre otras cosas. Por último, los usuarios también pueden dejar reseñas de otros, dando a conocer detalles sobre su puntualidad o compromiso, por ejemplo.

4. 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.

4.1. *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*.

4.2. *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.

4.3. *Rubocop*

Seguir alguna guía de estilo de código para *Ruby* monitoreado por la gema *Rubocop*. Se les subirá una configuración básica opcional a seguir. La configuración de estilo queda a decisión de cada grupo, pero una vez fijadas deben respetarse.

4.4. *RSpec*

Escribir tests para la aplicación utilizando la plataforma de testing en *Ruby* llamada *RSpec*. Una vez escritos los tests, estos **deben** pasar. La aplicación no puede tener tests fallando.

4.5. *Heroku*

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

5. 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*.

5.1. *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*.

5.2. Coevaluación

Por cada entrega deberá responderse una coevaluación de sus compañeros. Esta nos servirá para poder conocer el estado del grupo respecto al trabajo que cada uno está realizando y analizar si debemos o no tomar alguna medida al respecto. Detalles de esta se especificarán luego de la primera entrega.

5.3. 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.

5.4. Sprints

En total, son 4 Sprints. Cada sprint se realiza mediante su repositorio asignado de grupo en la organización de *GitHub* del curso, donde se corregirá el último *commit* en la rama *main* dentro de plazo. Luego del sprint 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, algunos sprints incluyen un aspecto obligatorio o evaluación específica a realizar. A continuación, se listan a grandes rasgos los sprints:

5.4.1. Sprint 0 (8 de abril)

Relatos de usuario y aplicación mínima “Hello World!” publicada en *Heroku* con una configuración de *Rubocop*.

5.4.2. Sprint 1 (29 de abril)

Modelación mediante diagrama E/R de la aplicación y funcionalidades negociadas con el *Product Owner*.

5.4.3. Sprint 2 (20 de mayo)

Funcionalidades negociadas con el *Product Owner*.

5.4.4. Sprint 3 (10 de junio)

Funcionalidades negociadas con el *Product Owner*.

5.5. Presentación Final

Finalmente, luego de los sprints 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.

5.6. Nota

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

- \bar{E}_P : Promedio de notas de sprints parciales.
- R_F : Nota de revisión final, como producto desarrollado.
- P_F : Nota de presentación final.

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

$$P = 0.5 \cdot \bar{E}_P + 0.2 \cdot R_F + 0.3 \cdot P_F$$

5.6.1. Premio mejores proyectos

Al final del semestre, el equipo docente seleccionará a aquellos grupos que consideren que desarrollaron los mejores proyectos.

En el caso de que su presentación final cumpla con los requisitos estipulados en dicho momento, el grupo podrá obtener un 1 punto extra en su nota final del proyecto.

6. 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 el *Sprint 0* a pesar que no terminen siendo todos finalmente implementados.

7. 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 Académica 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.