PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2113 — Diseño Detallado de Software

Proyecto Semestral - Entrega 1

Entrega

- **Fecha y hora:** Viernes 1 de abril de 2022, 13:00
- **Lugar:** Cuestionario en Canvas

Objetivos

- Analizar un problema medianamente complejo enfocado a la creación de un producto tecnológico
- Modelar la solución del problema identificado y utilizar diagramas para comunicar la solución planteada

1. Descripción

Para esta entrega, cada estudiante deberá realizar la etapa inicial del diseño de un sistema de software. Para este propósito, se permite elegir entre distintas opciones de proyecto según el *stack* de tecnologías que preferiría utilizar durante el proyecto semestral. **El *stack* que se utilice en esta entrega podría no ser el mismo que utilice durante el resto del proyecto semestral, pero sí recomendamos elegir según sus preferencias**.

Cada *stack* de tecnologías está asociado a un tema específico, pensado de manera que la elección del *stack* haga sentido para la resolución del problema. La descripción detallada de cada tema se encuentra en el apéndice de este enunciado:

- **Rails** (*back-end*) y **React** (*front-end*): Especificaciones para Rails & React
- Rails (*back-end*) y **Unity** (*front-end*): Especificaciones para Rails & Unity
- Rails (*back-end*) y HTML + CSS puro (*front-end*): Especificaciones para Rails & HTML + CSS puro

En específico, en esta entrega lo que se busca es que diseñe la estructura base del sistema y que sea capaz de comunicar sus ideas por medio del uso de diagramas UML. Se esperan al menos 5 diagramas UML, uno por cada una de las vistas del Modelo 4+1. Además deberá indicar a cuál vista del Modelo 4+1 responde cada diagrama UML entregado y argumentar (brevemente) su decisión.

Para realizar preguntas sobre esta entrega o sobre el proyecto en general se deberá utilizar el repositorio oficial del curso en GitHub: **IIC2113/Syllabus**.

2. Entregables

- Informe en PDF o Markdown con los diagramas UML del proyecto, indicando de manera clara a cuál vista del Modelo 4+1 corresponde cada uno
- (Opcional) Documento con supuestos e información adicional relacionada con los entregables

3. Distribución de Puntaje

- (20 %) Diagrama(s) UML para la Vista Lógica
- (20 %) Diagrama(s) UML para la Vista de Procesos
- (20 %) Diagrama(s) UML para la Vista de Implementación
- (20 %) Diagrama(s) UML para la Vista Física
- (20 %) Diagrama(s) UML para la Vista de Escenarios

Para cada vista se espera un diagrama UML, pero si se entrega más de uno se corregirán todos y se utilizará el promedio de sus puntajes.

Apéndices

A. Especificaciones para Rails & React

Este proyecto contempla la creación de un sitio web de *ecommerce* en el que se pueda ver una lista de productos disponibles para comprar, agregar productos al carrito de compras, y luego finalizar la compra de los productos del carrito.

Las funcionalidades mínimas que deben estar presentes en la aplicación son la siguientes:

Autenticación de usuarios Debe existir un sistema básico de autenticación de usuarios. Será requisito estar autenticado para realizar una compra, ya que cada una de estas quedará registrada a un usuario en particular. Debe también existir un rol de administrador que gestione los productos disponibles, su información, sus categorías y los cupones disponibles.

Lista de productos Se debe poder visualizar la lista de productos disponibles y su información asociada.

Categorías de productos Cada producto debe estar asociado a una categoría. Las categorías son de libre elección y gestionadas por el rol de administrador.

Carrito de compras Se debe poder agregar productos a un carrito de compras, pudiendo visualizar la lista de productos añadidos al carrito, el valor de cada uno y el total de todos los elementos dentro de este. También se debe poder modificar la cantidad de un determinado producto dentro del carrito y este debe ser persistente entre sesiones del usuario (e.g. acceder al mismo carrito desde distintas ventanas del navegador o distintos dispositivos en los que se inicie sesión).

Cupones Deben existir cupones de descuento que se pueden aplicar sobre los productos. Estos pueden ser de dos maneras. Existen cupones que pueden ser aplicados sobre todos los productos, independiente a su categoría (sobre el valor total del carrito). Pero también existirán aquellos que solo podrán aplicar descuentos sobre alguna categoría en específico, sin afectar el precio de aquellos productos que no se encuentran en dicha categoría.

Finalización de pedido (checkout) Se registra en el servidor la compra y esta queda en el historial de compras del usuario. Para este paso no es necesario simular el ingreso de información de pago ni similar, solo basta con un botón o acción que gatille este evento. El carrito de compras debe quedar vacío posterior a esta acción.

Historial de compras Visualización del historial de compras realizadas por el usuario. Cada pedido debe ser distinguible de otro y por cada uno se deben mostrar los productos, el valor de cada uno, el valor total de la compra y los descuentos (cupones) aplicados en el pedido en cuestión.

B. Especificaciones para Rails & Unity

Para este proyecto se deberá desarrollar un centro de videojuegos (**INSERTE NOMBRE AQUÍ**), donde los usuarios podrán acceder a diferentes tipos de juegos y ganar monedas o logros para aumentar su nivel de jugador en el centro de videojuegos.

Modelos

- **User**: Modelo enfocado en el usuario, ayudará a identificarlo y contendrá la siguiente información:
  - ID
  - Email
  - Nombre de usuario
  - Nombres
  - Apellidos
  - Contraseña
- **Game**: Modelo para identificar los juegos disponibles en la plataforma, contendrá la siguiente información:
  - ID
  - Nombre
  - Descripción
  - Categoría
  - Dificultad
  - Precio
- **Match**: Modelo que identifica cada partida que el usuario realice en un juego en particular, contendrá la siguiente información:
  - ID
  - User ID
  - Game ID
  - Tiempo de juego
  - Puntaje
  - Monedas obtenidas
  - Estado de partida (iniciada, ganada, perdida)
- **Wallet**: Modelo que permite mantener un estado de ganancias para el usuario, con el monedero podrá acceder a juegos y contendrá la siguiente información:

    - ID
    - User ID
    - Monedas disponibles

- **Achievement**: Modelo que permitirá al usuario obtener diferentes niveles en la plataforma. Con los logros obtenidos podrá acceder a beneficios en monedas y en el ranking. Contiene la siguiente información:

    - ID
    - User ID
    - Match ID
    - Descripción del logro
    - Monedas Bonus
    - Nivel (importancia del logro)

Juegos

A continuación, se enumeran las distintas opciones de juegos a implementar en el centro de videojuegos:

- **Atrapar objetos**: La finalidad de este juego es conseguir puntos atrapando objetos, ya sea monedas, frutas, etc. El personaje puede moverse en vertical y/o horizontal. Por cada artículo que atrape recibirá un punto de juego, a menos que se implementen varios tipos de artículos con diferente ponderación.
- **Atacar enemigos**: La finalidad de este juego consiste en vencer a algún enemigo, ya sea a través de ataques directos (espada) o ataques a la distancia (space invaders). Por cada enemigo vencido se ganará un punto de juego y algunos enemigos podrían dejar atrás recompensas extras como monedas.
- **Evadir obstáculos**: La finalidad de este juego es evadir obstáculos la mayor cantidad de tiempo posible. Cuando tocas un obstáculo pierdes. Por ejemplo, el juego de dinosaurio que se despliega cuando no tienes internet en chrome, cuya finalidad es evadir cactus saltando. Ya sea por obstáculo evadido o por tiempo alcanzado, deberás asignar un puntaje al jugador.
- **Laberinto**: La finalidad de este juego es atravesar diferentes laberintos llegando desde un punto A hasta un punto B. Se tiene obstáculos como murallas, pero el tocarlos no implica el final del juego. El juego debe generar infinitos niveles de forma aleatoria y por cada nivel sorteado o por el tiempo que tomó realizarlo se debe asignar puntaje. También se puede dejar recompensas en monedas a lo largo del laberinto.
- **Saltos infinitos**: En este juego estilo Doodle Jump, se debe implementar un personaje que irá saltando sobre plataformas que irán apareciendo aleatoriamente a medida que sube, donde por cada plataforma que pise recibirá un punto de juego. También deberán haber monedas extras que podrá atrapar a lo largo del camino.

C. Especificaciones para Rails & HTML + CSS puro

Para este proyecto se deberá desarrollar un Planner de cursos (similar al de Siding **https://planner.ing.puc.cl/**). Este consistirá en implementar tanto la administración de este como su uso para les estudiantes.

- **Usuario**
  - Deberá existir un registro y login en la *landing page*.
  - Debe existir por lo menos dos roles: administrador y alumno.
  - El alumno deberá poder ver sus cursos aprobados y a partir de esto, visualizar cuál será su malla para los próximos semestres.
- **Curso**
  - Se debe poder crear cursos.
  - Estos deben tener los atributos: nombre, descripción, nrc y semestre en que se realiza el curso.
  - Cada curso tendrá requisitos, los cuales podrán ser modificados por un usuario administrador (ej: Calculo II tiene como requisito Cálculo I).
- **Carrera**
  - El administrador debe tener acceso a crear nuevas carreras, las cuales consisten en un conjunto de cursos que el alumno debe cursar para poder graduarse de ella.
  - Los alumnos deberán poder inscribirse a esta.
  - El sistema deberá soportar la opción de que una carrera tenga un major y/o minor.