# Escuela Colombiana de Ingeniería
# Arquitecturas de Software – ARSW
# Proyecto de curso – Sprint 3
# 2020 - 2

Las siguientes son las condiciones de evaluación del Sprint 3 para el proyecto de ARSW.

## Atributos No Funcionales

Usando el modelo de documentación de atributos no funcionales, para los siguientes atributos de calidad documente 2 escenarios. Usted debe ser capaz de demostrar el cumplimento de dichos escenarios usando los diagramas de arquitectura, una prueba funcional, un experimento, una prueba de carga, un vídeo pre-grabado etc., el hecho es que demuestre que la arquitectura propuesta para su proyecto se capaz de asegurar el escenario no funcional documentado.

- Usabilidad
- Escalabilidad
- Disponibilidad
- Rendimiento (Performance)

## Arquitectura Backend, extendiendo Cache

La arquitectura del *backend* del proyecto debe separar claramente las capas de control, lógica, caché, persistencia, y garantizar un bajo acoplamiento entre las mismas. La capa de 'caché', a diferencia de la de persistencia, es responsable de mantener sólo información volátil requerida por la aplicación (por ejemplo, el estado de una partida, si se tratara de un juego).

La inclusión de una capa de control de cache, como estrategia para mejorar la disponibilidad y rendimiento (performance del sistema)

## Arquitectura Frontend, extendiendo Real Time

De la misma manera, el *frontend* debe estar correctamente modularizado y con un diseño por capas. Debe ser plenamente visible la capacidad del sistema para funcionar bajo un esquema de *Real Time,* donde el usuario reciba retroalimentación de la plataforma sin que el mismo deba interactuar de manera directa con el sistema.

## Despliegue en Heroku y Manual de Uso

Como primero se hará una revisión -sin participación del equipo de desarrollo- de lo que esté desplegado en la nube, se debe tener cuidado de incluir en el archivo README.md del repositorio, usando formato Markdown:

- Nombre, asignatura,  descripción, autores.
- Enlace a la aplicación en Heroku.
- Arquitectura de la aplicación especificando:
    - Diagrama de clases
    - Diagrama de Entidad relación
    - Diagrama de despliegue
    - Diagrama de casos de uso

    - Diagrama de Componentes
- Indicaciones básicas de uso de la aplicación (usuarios disponibles por defecto, instrucciones de manejo, etc).

## Análisis estático de código y Pruebas

- Registrar el repositorio en el sistema de análisis estático de calidad de código CODACY: https://www.codacy.com
- El proyecto debe realizarse sobre una línea de producción que contemple TDD, Integración, y Entrega Continua. Esto implica:
    - Hacer un desarrollo dirigido por pruebas (por ahora, de la capa lógica), en donde se harán especificaciones de lo que se implementará, y sobre éstas se hará un diseño basado en clases de equivalencia y condiciones de frontera para las pruebas. Las pruebas son indispensables principalmente si hay modelos que contengan reglas de la aplicación de lado del servidor.

## Completitud Funcional

Se evaluará la completitud funcional del sistema, tanto las historias de usuario colaborativas como las no colaborativas, haciendo especial énfasis en el cumplimiento de los atributos no funcionales.