Seminario inicial

Objetivo

Introducir al alumno en la resolución de un problema de diseño de sistemas, entendiendo cuál es su alcance y cómo llevar a cabo su solución, integrando conocimientos previos (estudiados en las materias correlativas) y mostrando el contenido que van a aprender en nuestra asignatura.

Motivación

La existencia de un déficit en el análisis sobre la problemática que subyace en un dominio planteado, lo que lleva a que en las etapas posteriores de la resolución, donde se necesita ir bajando el nivel de abstracción hasta llegar a tener la solución concreta, se produzcan errores u omisiones que hacen que no termine siendo efectivo o eficiente.

Temas

- Presentación de la Cátedra
- Definición de diseño
- Relación con conocimientos de materias correlativas
- Ejercicio para modelar
  - Explicación método
  - Ejecución método
- Introducción a la arquitectura
  - Aspectos
  - Atributos de calidad

Contenido

Presentación de la Cátedra

Cátedra unificada

Desde el año 2016 la cátedra se ha unificado y se encuentra trabajando en conjunto todos los temas referidos al aspecto académico.
Para este trabajo conjunto se han definido algunos lineamientos en común:

- ***Programa Unificado***
  El programa de la materia se encuentra dividido en 8 unidades temáticas, a saber:
  - Diseño y Sistemas
  - Herramientas de Concepción y Comunicación del Diseño
  - Diseño con Objetos
  - Diseño de Interfaz de Usuario

- Diseño de Datos y Estrategias de Persistencia
- Introducción al Diseño de Arquitectura
- Validación del Diseño
- Diseño y Metodologías de Desarrollo

Así mismo, el programa completo se encuentra en el siguiente link.

- ***Régimen de aprobación directa***
  El régimen de promoción de la cátedra se encuentra publicado en el siguiente link.

- ***Trabajo Práctico Anual Unificado***
  El trabajo práctico ayuda a articular los temas teóricos vistos en la cursada y su aplicación en un sistema de información.
  Este trabajo tiene la particularidad de ser iterativo e incremental para poder acompañar en el tiempo las distintas temáticas que se ven en la cursada.

- ***Seminarios de Tecnologías***
  Este año contaremos con 6 seminarios tecnológicos. Los mismos se realizarán los días sábados en las fechas que se detallan debajo, en dos turnos: 9 y 14 hs.
  - 11/4 - Java, Git y Maven
  - 16/5 - API Rest
  - 13/6 - Maquetado Web
  - 8/8 - ORM/Hibernate
  - 29/8 - Arquitectura Web
  - 7/11 – Temática a confirmar

- ***Final Unificado***
  Todos los finales, desde el año en que la cátedra se unificó, se encuentran publicados en la carpeta pública del Drive (link).
  En general no encontrarán resoluciones de los exámenes finales publicadas. Esta decisión está basada en que no existe una solución única para una problemática vinculada al Diseño, ya sea para Sistemas de Información u otra disciplina. Es por ello que los invitamos a ustedes a resolverlos, generar una propuesta de solución, justificar sus decisiones de diseño y luego enviarnos la propuesta a través de las Redes Sociales. A partir de ello generaremos en un intercambio con ustedes, para ir iterando esa solución y llegar a un diseño que cumple con los requisitos y contiene las cualidades de diseño esperadas.

- ***Clases de Consulta para Finales***
  Desde Diciembre 2018 se han implementado las clases de consulta para Finales. Las mismas son dictadas en la sede Medrano, con fechas y hora a confirmar antes de cada llamado a final.

- ***Comunicación a través de Redes Sociales***
  Desde 2018 abrimos nuestras redes sociales para poder estar más cerca de los alumnos y transmitir más rápidamente los comunicados oficiales de la cátedra.
  - Facebook: ddsutnbaoficial
  - Instagram: ddsutnbaoficial

Días de cursada

- Martes Mañana y Noche
- Miércoles Mañana y Noche
- Jueves Mañana
- Viernes Noche
- Sábado Mañana

Contacto

Cualquier duda o consulta referida a la cátedra enviar un mail a la coordinación: Esp. Ing. Lucas J. Saclier - saclier@gmail.com

Definición de Diseño

Primero conocer la definición formal de Diseño:

- (sustantivo) una **especificación** de un **objeto**, realizada por un **agente**, destinada a cumplir **objetivos**, en un **contexto**, usando un conjunto de **componentes primitivos**, satisfaciendo **requerimientos**, sujeto a **restricciones**.

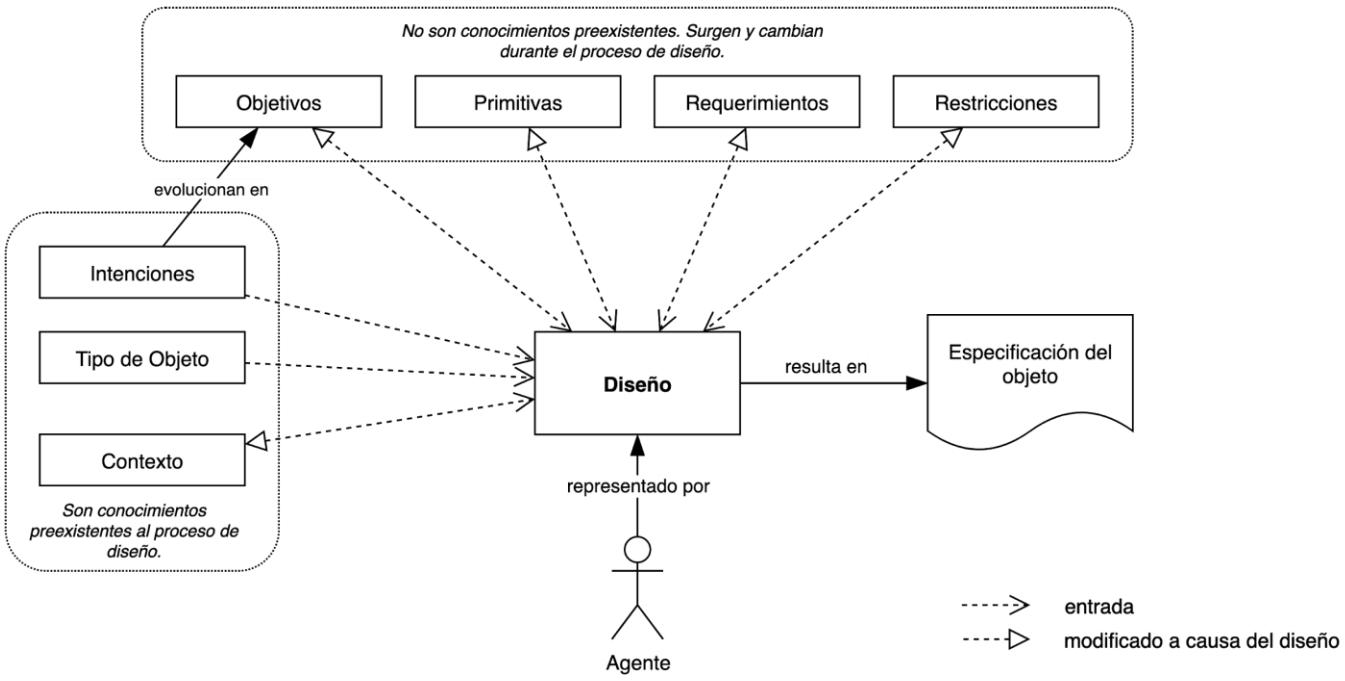
- (verbo) crear un diseño, en un contexto (dentro del que está inmerso el agente).

Ahora hay que entender que significan los conceptos que incluye la definición:

Concepto	**Significado**	**Aplicación**
Especificación	Una descripción detallada de un objeto en términos de su estructura (ej: cuales son sus componentes y cómo se relacionan). La especificación puede ser puramente mental, o dada en una representación simbólica, o provista por un modelo físico, o manifestada como el objeto mismo. La naturaleza de la especificación va a depender de la clase del objeto a diseñar.	Usamos una representación simbólica (UML).
Objeto	Una entidad a ser diseñada, no necesariamente fisica.	Es el artefacto dentro del "dominio de aplicación" que es diseñado para poder llevar a cabo las actividades de ese mismo. En nuestro caso va a ser un software. Un dominio de aplicación comúnmente hace referencia a un entorno organizacional, como un negocio o una parte de él.
Agente	Quien/es realiza/n la especificación del objeto.	Estos van a ser ustedes, los diseñadores.
Contexto	El ambiente o escenario en donde el objeto va a existir y operar.	El entorno externo al dominio de aplicación.

Objetivos	El impacto a generar por parte del objeto sobre el contexto.	
Primitivas	El conjunto de elementos que componen al objeto.	En nuestro caso son los del paradigma de objetos, patrones de diseño, ...
Requerimientos	Las propiedades estructurales o de comportamiento que debe tener el objeto a diseñar. Una propiedad de comportamiento define cómo responde a un estímulo del ambiente. Una propiedad estructural es una cualidad que el objeto debe poseer más allá de las condiciones del ambiente.	
Restricciones	Es una limitación de la estructura o el comportamiento sobre el objeto a diseñar.	Un ejemplo para un sistema software sería el de utilizar tecnologías open-source.

Y saber de qué forma se relacionan entre sí a través de este diagrama conceptual:



Por último, una definición “aplicada” al Diseño de Sistemas, citaremos a James Rumbaugh en su libro Modelado y Diseño Orientado a Objetos, la cual nos dice:

“Es la estrategia de alto nivel para resolver problemas y construir una solución. Éste incluye decisiones acerca de la Organización del sistema en subsistemas, la asignación de subsistemas a componentes de hardware y software, y decisiones fundamentales conceptuales y de política que son las que constituyen un marco de trabajo para el diseño detallado.”

Relación con conocimientos de materias correlativas

Tal como el programa de la materia lo menciona, es necesario tener regularizadas Análisis de Sistemas y Paradigmas de Programación para poder cursar.

Siguiendo con los conceptos que se desprenden de la definición de diseño, una vez que los definimos podemos relacionarlo con el contenido de las asignaturas:

- Análisis de Sistemas => objetivos, requerimientos, restricciones y contexto.
- Paradigmas de Programación => componentes primitivos
- Diseño de Sistemas => decisiones en base a principios de diseño a alto (externo) y bajo nivel (interno), especificación y comunicación de la solución.

Ejercicio para modelar

1. Leer relevamiento sobre dominio

Objetivo: Organización de campeonatos de futbol virtual para grupos de personas.

Dominio

La organización de un campeonato abarca desde la creación del torneo, la registración de los equipos, el fixture partidos, hasta la definición de los ganadores.

Un torneo es creado por un usuario anónimo que le define un nombre, una clave de acceso para la administración y una cantidad de equipos. Al crearlo de forma exitosa el sistema le asigna un identificador al torneo para su posterior administración.

Todo usuario que conozca el identificador y la clave de acceso podrá administrar el torneo. Cualquier usuario que conozca solo el identificador podrá visualizar el estado del torneo.

Un administrador registra los equipos con un nombre. Es válido agregar o quitar equipos hasta antes de que se marque el inicio del torneo.

Un administrador marca el inicio del torneo. Esto es posible siempre y cuando haya registrados la cantidad de equipos previamente definida. Ahí mismo se genera el fixture.

Un fixture se crea a partir de una modalidad, que por el momento hay 2 definidas: 1) todos contra todos solo ida; 2) todos contra todos ida y vuelta. Un fixture se compone de fechas y las fechas de partidos. En una fecha juegan todos los equipos.

A medida que se van jugando los partidos, los protagonistas deben informarle al administrador el resultado con evidencia del mismo (pueden tomar una foto de las estadísticas finales del partido), para que luego éste cargue el resultado del encuentro disputado y el dia y la hora en que sucedió efectivamente. Las fechas se tiene que jugar en el orden que define el fixture.

Un participante puede ver la tabla de posiciones del torneo hasta el momento y el fixture.

En una tabla de posiciones se listan todos equipos ordenados por puntos acumulados, partidos jugados y diferencia de goles. Los puntajes se asignan de la siguiente manera: 3 puntos partido ganado, 1 punto

empate, 0 puntos partido perdido.

Al completarse todas las fechas o al momento en que se decida terminar el torneo por parte del administrador, aquel equipo que se encuentra primero en la tabla de posiciones es el campeón.

Se mantiene informado a los participantes sobre novedades del torneo siempre y cuando éstos se hayan suscrito al servicio de notificación. Son consideradas novedades el inicio y finalización del torneo, los próximos partidos a jugar y resultado de partidos jugados. El servicio es provisto por un tercero que dado un contacto (ej: email, nro teléfono, ...) y tópicos de interés (que en este caso son ids de torneos y nombre de equipo), lo registra y espera recibir novedades de los tópicos para luego hacerlas llegar por el medio correspondiente a los interesados.

2. Entender qué hay que resolver

Para comenzar a diseñar, basándonos en el diagrama conceptual de la definición, lo primero que debemos conocer son las “entradas” a nuestro diseño.

- El contexto para saber donde va a operar el artefacto, cuales estímulos recibirá, y por parte de quienes.

- Los objetivos para saber cual es el efecto sobre el contexto de las acciones que se ejecutan, es decir, para saber el porque lo estoy haciendo. Entonces, si no conozco o no entiendo los objetivos no tengo motivación para desarrollar un sistema.

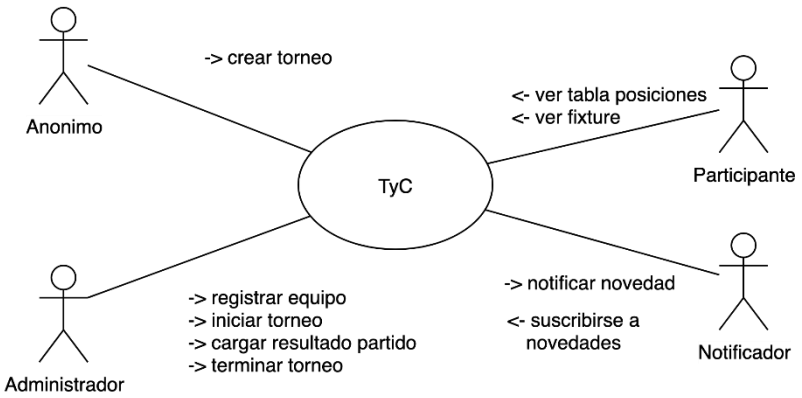
- Los requerimientos para saber qué propiedades debe tener el artefacto para cumplir con el objetivo, desde la perspectiva de comportamientos, es decir, las respuestas que espera el contexto, y desde la perspectiva estructural, es decir, de cómo tiene que estar construido para que encaje en el contexto.

- Las restricciones para saber que limitaciones necesitamos tener en cuenta cuando especifiquemos el artefacto.

Tenemos que “aterrizar” la idea de la solución, es decir, ir bajando el nivel de abstracción de forma gradual. La explicación de un dominio de aplicación por parte de un experto es el nivel más alto y en el otro extremo tenemos el “código”, las instrucciones en un lenguaje de programación que deben ser interpretadas por el computador. Es un salto muy grande el que hay que hacer, por lo tanto intentarlo de forma directa representa un riesgo muy alto de no resolver la problemática planteada.

Entonces, empecemos por describir el contexto. Para esto nos preguntamos donde está ubicado mi dominio en el contexto externo, con quienes interactuamos y qué esperan que resolvamos (dado que somos dueño de una entidad). Tratamos a nuestro artefacto como una caja negra porque no nos interesa saber dentro de él cómo va a cumplir, sino que necesita cumplir y para quién. Con esto nos va a quedar claro que está dentro y que queda fuera de nuestro alcance.

Diagrama de contexto



- La entidad principal es el “Torneo” y luego todo lo que se deriva de esta tal como “Fixture”, “Fecha”, “Partido”, “Resultado”, “Equipo”, “Modalidad Fixture”.

Luego, obtengamos un listado de los requerimientos que nos guíe en lo que esperan del comportamiento de nuestro artefacto los actores que se describen en el contexto.

Requerimientos Funcionales:

- Un anónimo puede crear un torneo con una modalidad por default o alguna de las existentes.
- Un usuario anónimo puede ser administrador del torneo conociendo el id y contraseña del mismo.
- Un usuario anónimo puede ser participante del torneo conociendo el id del mismo.
- Un administrador puede registrar/desregistrar equipos en un torneo.
- Un administrador puede dar comienzo y fin al torneo que administra.
- El sistema tiene que generar un fixture con una modalidad definida para un torneo.
- Un participante puede conocer el estado de un torneo: tabla de posiciones y fixture completo, fecha en curso y la próxima.
- Un participante puede suscribirse a novedades de un torneo y de equipos.
- El sistema debe notificar a los participantes sobre novedades que ellos marcaron de interés.

Atributos de calidad (fuera del alcance del seminario)

Luego, describimos lo que dentro del artefacto se debe hacer para cumplir con los requerimientos, es decir, describir las funcionalidades. Un medio para esto serian los CUs (Casos de Uso).

3. Encarar la resolución

Poder empezar a darle forma a nuestra solución, hay que describir un escenario, es decir, una secuencia concreta de ejecuciones de funcionalidades que van a llevar a cabo los actores. Para esto tenemos que tomar la entidad de la cual somos responsables y describir su ciclo de vida o la forma en que opera. Es importante concentrarse en el “camino feliz”, que sea el más representativo y que nos ayude a comunicar mejor la solución que tenemos en mente.

Armar este escenario lleva a primero tener que pensar que cosas de antemano tengo que tener cargadas/definidas antes de que un actor interactúe (ej: catálogos, algoritmos, configuración, ...) porque va

a hacer uso de ellas. Luego, empezamos a listar las funcionalidades, a modo de transacciones, que los distintos actores van a ejecutar en orden contra nuestro artefacto para recorrer el ciclo de vida de inicio a fin de la entidad principal. Por último, detectar cuales funcionalidades son ejecutadas como consecuencia de algún evento y no directo de un actor.

Analizando el dominio para extraer este escenario nos vamos a encontrar con descripciones que aportan detalle de una operación particular (ej: un cálculo, un procedimiento, restricciones, ...), o mismo detalle de una operación que está fuera de nuestro alcance, entonces es importante no enredarse con eso, y nos mantengamos enfocados en nuestro artefacto.

Escenario

1. El sistema define modalidades para generación de fixture y marca una default.
2. Un usuario anónimo dá de alta un torneo con una modalidad default y recibe un id de torneo.
3. Un usuario administrador registra un equipo para el torneo.
4. Los participantes se suscriben a las notificaciones para el torneo y a su equipo.
5. Un usuario administrador registra otro equipo para el torneo.
6. Un usuario administrador inicia el torneo.
7. El sistema crea el fixture del torneo porque se inició el torneo.
8. El sistema notifica novedades inicio de torneo y partido a jugar.
9. [Los participantes juegan el partido de la primera fecha]
10. Un usuario administrador carga el resultado del partido jugado para el torneo.
11. El sistema notifica el partido jugado.
12. El sistema notifica que la fecha terminó porque se jugaron todos los partidos.
13. El sistema finaliza el torneo porque detecta que se jugaron todas las fechas.
14. El sistema notifica finalización del torneo.
15. Un participante consulta estado de torneo.

Entonces, hay que empezar a recorrer el escenario en orden. A medida que se avanza se contrasta con los requerimientos, descripciones de las funcionalidades, y las definiciones del dominio.

Como vamos usar el paradigma de objetos para dar solución, para nuestro modelo necesitamos detectar abstracciones, adjudicarles a cada una una responsabilidad, y hacer que conversen entre sí para resolver la funcionalidad. Por lo que las abstracciones deben tener una interfaz a través de la cual se puede interactuar con ellas y mantendrán un estado interno por medio de propiedades para poder seguir la conversación.

4. Armar un modelo

Un modelo es una simplificación de la realidad (en nuestro caso dominio de aplicación) y el propósito de construirlo es comprender mejor el sistema que estamos desarrollando y comunicar la estructura deseada y el comportamiento que tendrá.

Un buen modelo incluye aquellos elementos que tienen una gran influencia y omite aquellos elementos menores que no son relevantes para el nivel de abstracción buscado. Hay que tener en cuenta a quien se le va a presentar el modelo.

Los lenguajes de programación no tienen un nivel suficientemente alto de abstracción para facilitar una discusión sobre el diseño.

Para que la comunicación de la solución sea efectiva, es necesario mostrar desde distintas perspectivas. Un modelo puede ser estructural destacando la organización del sistema o puede ser de comportamiento, resaltando su dinámica. Cada modelo tiene que ser semánticamente cerrado (que pueda entender lo que representa sin necesidad de consultar por fuera de él). Esta comunicación va a ser a través del lenguaje el Unificado de Modelado (UML).

En la cursada estudiaremos UML, un lenguaje gráfico estándar para visualizar, especificar y documentar los artefactos de un sistema que involucre gran cantidad de software.

A medida que vayamos completando el modelo es necesario validar que los requerimientos se cumplen y que a su vez se respetan los principios que guían a un buen diseño. Estos principios serán estudiados a lo largo de la cursada.

Diagrama de clases primera iteración hasta paso 5

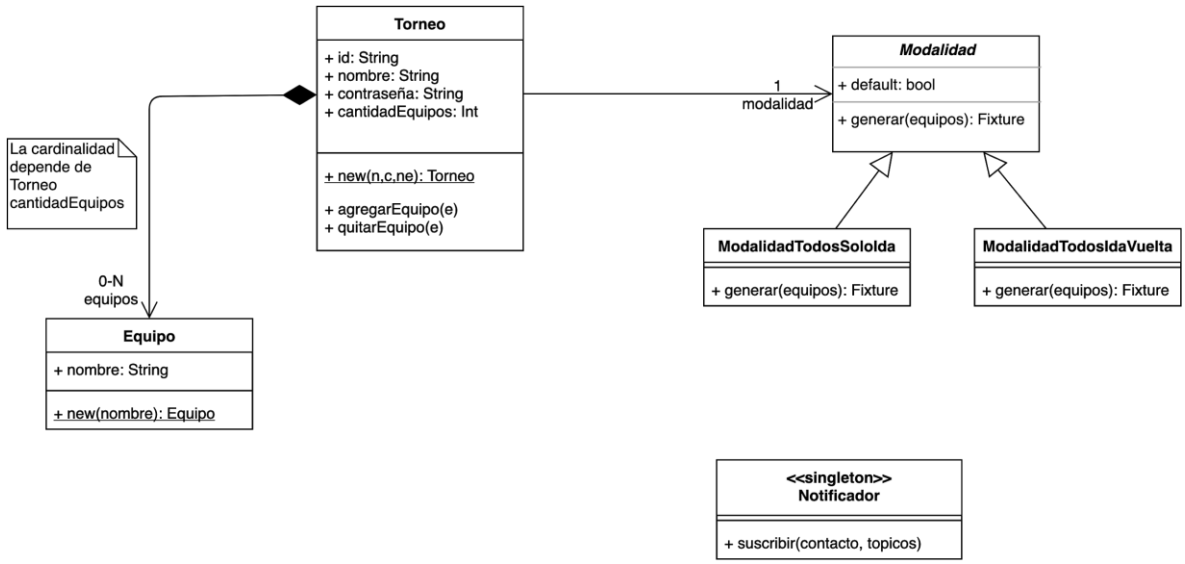
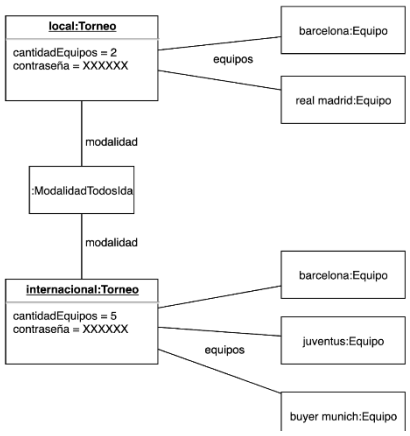


Diagrama de objetos primera iteración

Torneos creados con equipos registrados



Ahora continuemos completando el negocio con los siguientes pasos y lo hacemos “encajar” en el modelo planteado, es decir, es iterativo e incremental nuestro enfoque.

Quizás surja la necesidad de crear y/o replantear algunas abstracciones o las responsabilidades asignadas. Esto es normal dado que al construir el modelo se comprende mejor el sistema y la misma solución planteada, entonces muchas veces descubrimos oportunidades para la simplificación, la reutilización o reorganización.

Diagrama de clases para segunda iteración hasta paso 8

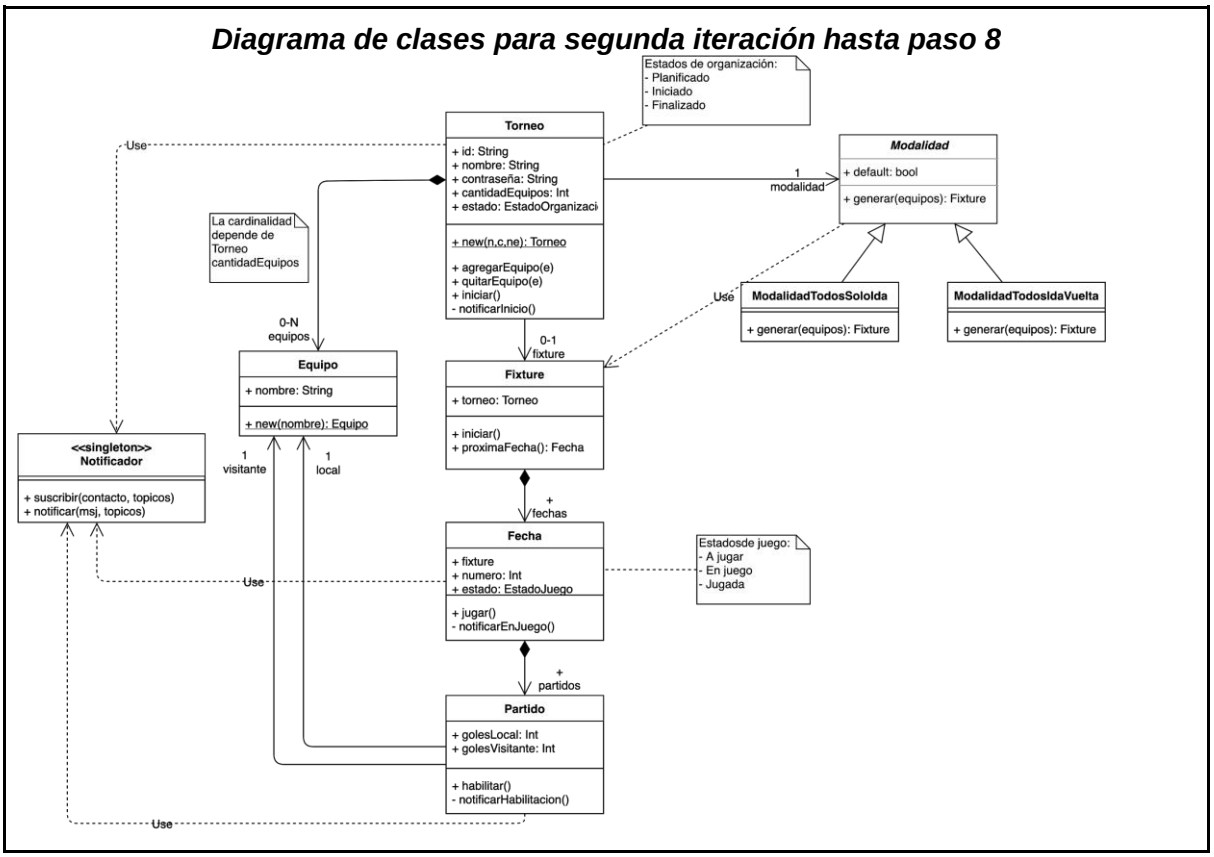
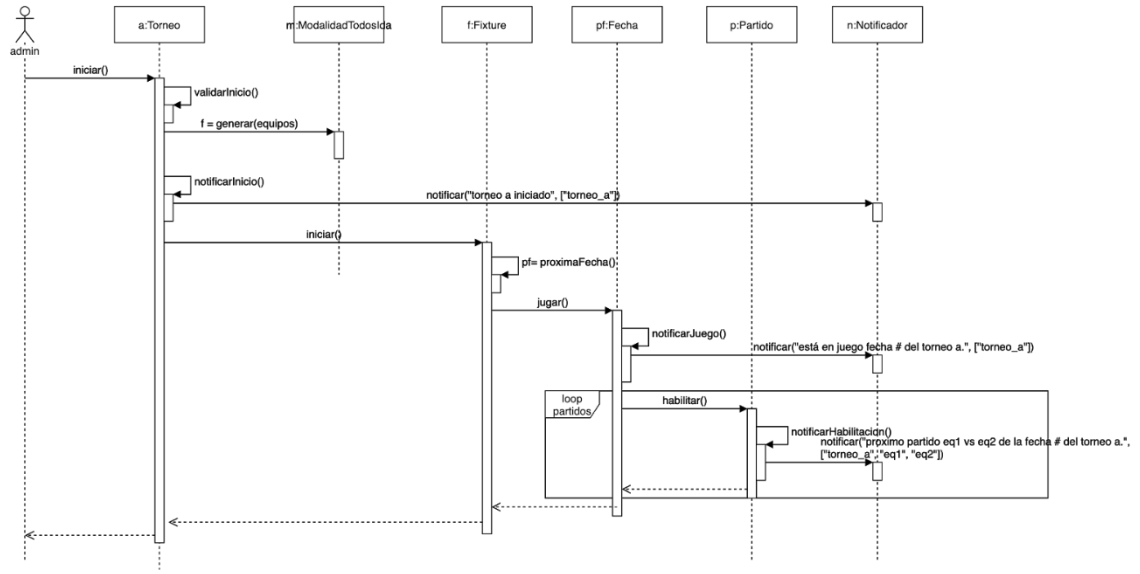


Diagrama de secuencia para segunda iteración hasta paso 8

Inicio de torneo A



Otra iteración más:

Diagrama de clases hasta el paso 13

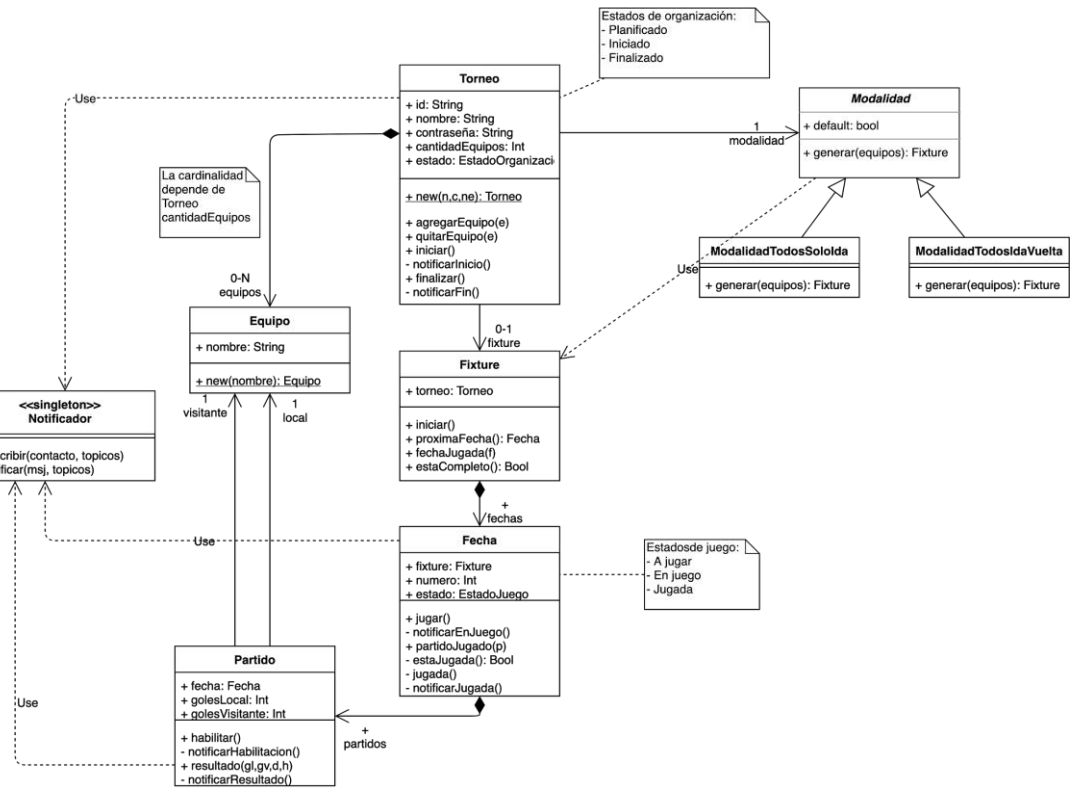
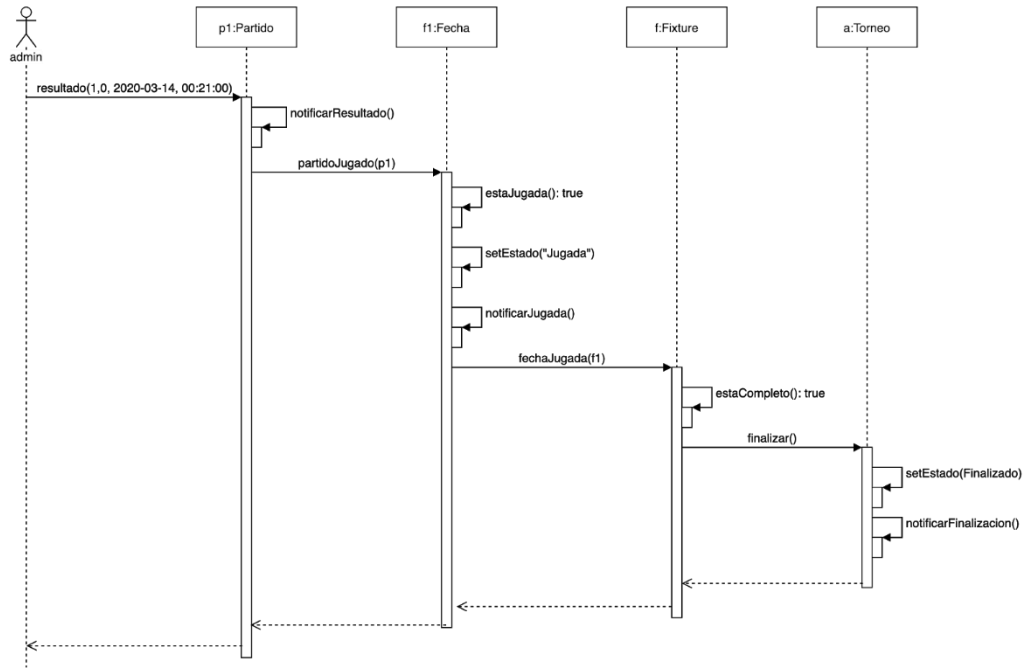


Diagrama de secuencia hasta el paso 13

Carga resultado de ultimo partido del torneo A



En la última iteración para completar el escenario:

Diagrama de clases final

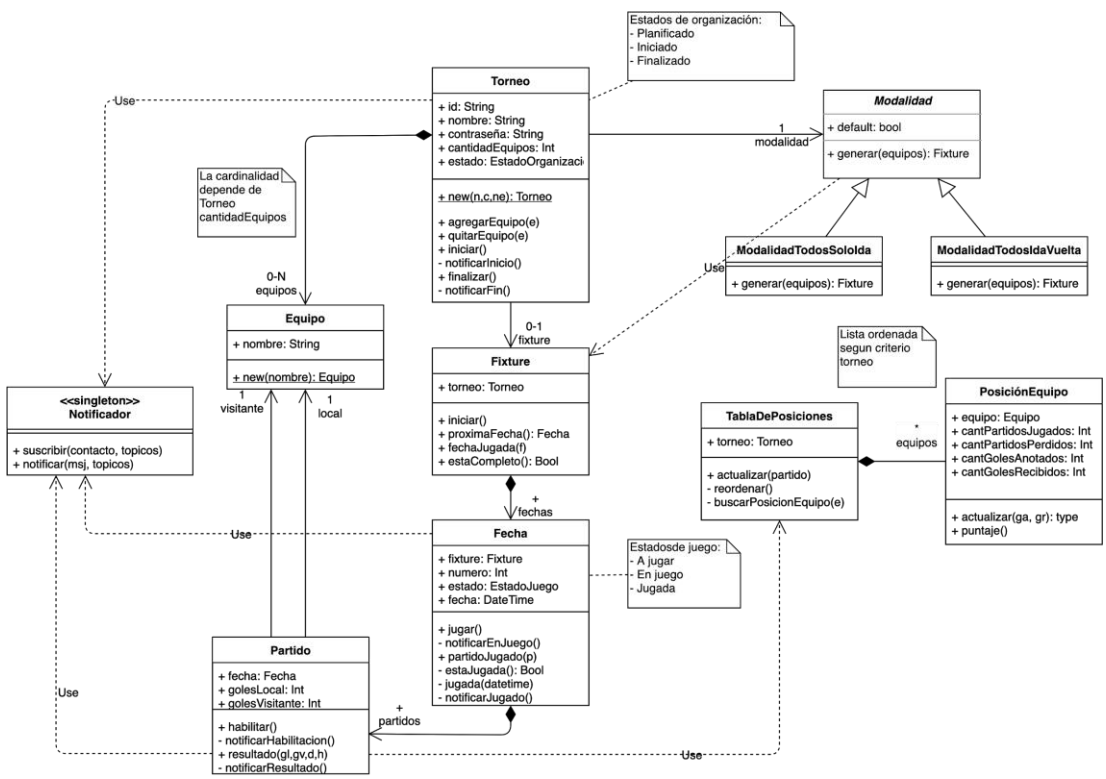
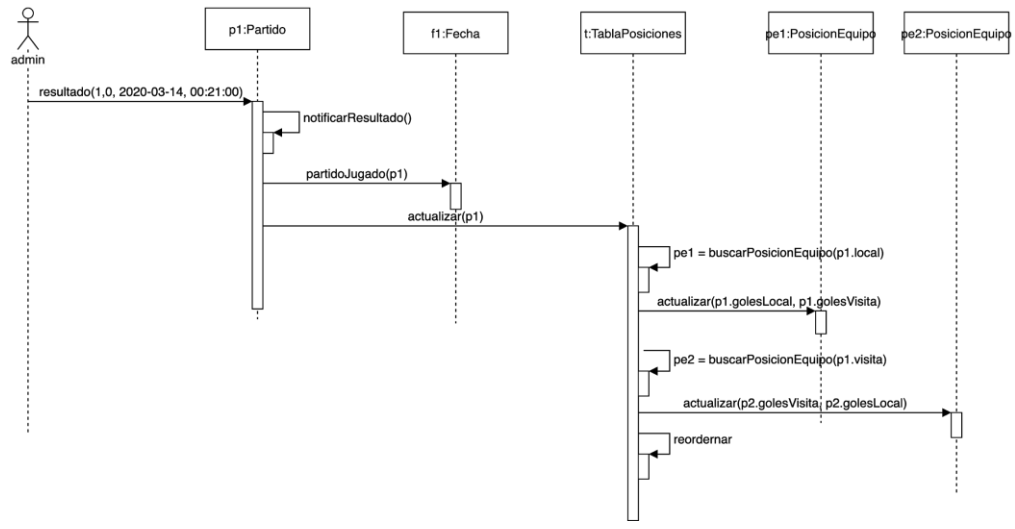


Diagrama de secuencia para el paso 14



Introducción a la arquitectura

Un sistema no es solo el dominio, hay otros **aspectos (*concerns*)** que no pasan por lo funcional y que debemos tener en cuenta a la hora de diseñar. Acá es donde entra en juego la arquitectura como parte del diseño.

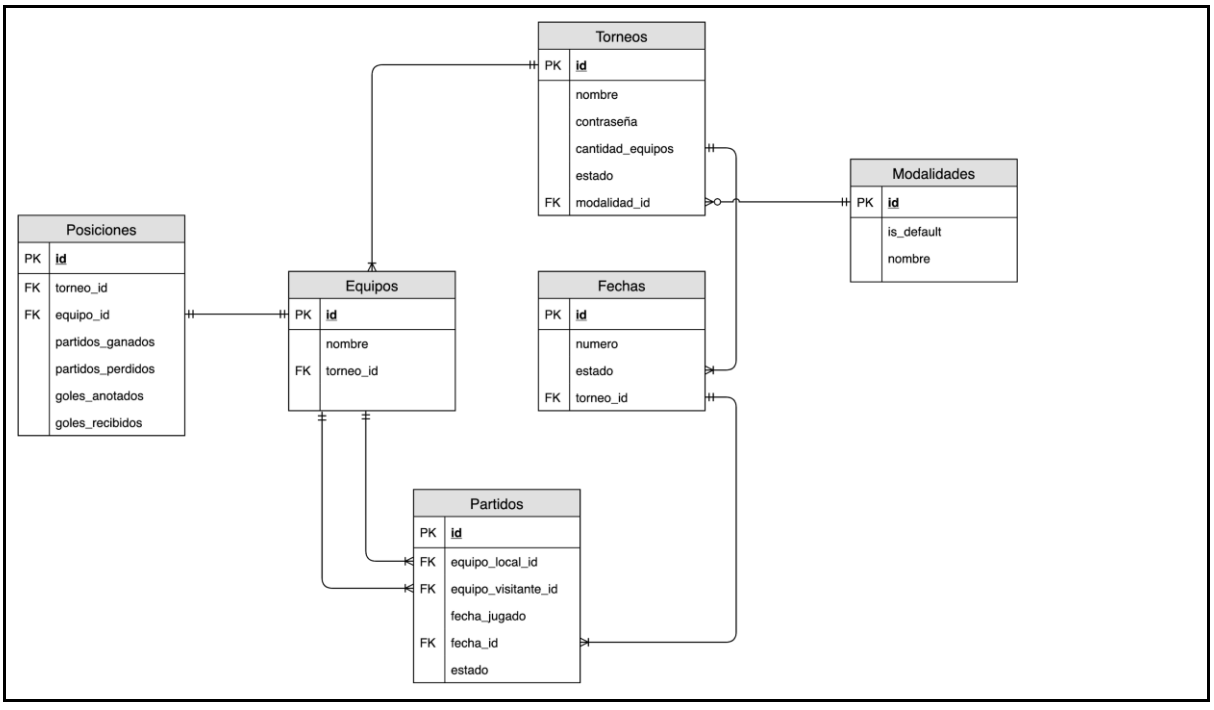
Algunas, y de las más importantes, cuestiones que debemos tener en cuenta a la hora de diseñar una arquitectura son:

- Requerimientos Funcionales
- Requerimientos No Funcionales (Vinculados a los Atributos de Calidad)
- Restricciones: de Negocio, técnicas
- Futuros Requerimientos
- Experiencia del Arquitecto
- Estilos y Patrones Arquitectónicos

En particular, existen 3 aspectos sobre los cuales más vamos a profundizar en la cursada que son:

- **Interfaz de usuario** se ocupa de la interacción con el usuario, es decir, mostrarle información al usuario y permitirle tomar acciones sobre esa información, modificarla, navegar entre las diferentes vistas o actividades, etc.

- **Persistencia** se ocupa de garantizar que la información del sistema perdure en el tiempo.



- **Seguridad** se ocupa de preservar la integridad, disponibilidad, y confidencialidad de los recursos del sistema (incluyendo hardware, software, firmware, datos, y la telecomunicación).

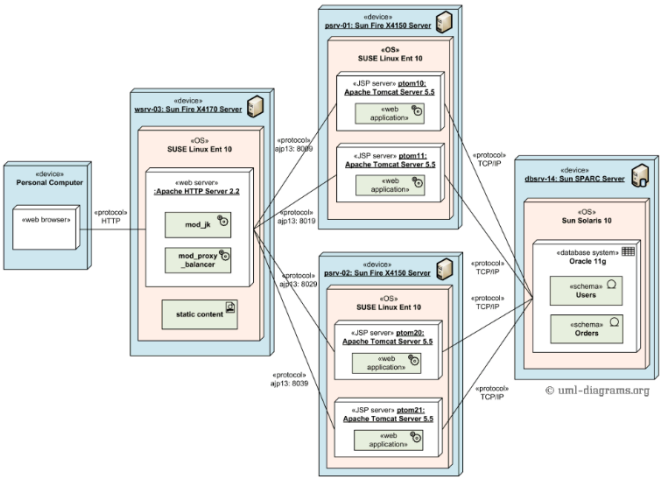
Y utilizamos **atributos de calidad** para representar la calidad con que van a ser cumplidos los requerimientos no funcionales o la aplicación en sí desde diferentes aspectos.

- Rendimiento
- Disponibilidad
- Seguridad
- Mantenibilidad
- ...

Existen 2 planos en los que podemos ver a la arquitectura:

- **Físico** que abarca cuestiones de hardware, despliegue de los artefactos, ambientes, red, y el stack tecnológico.

  Ejemplo de un despliegue



- **Lógico** que se refiere a la organización de los componentes, sus responsabilidades y como son las relaciones entre ellos.

  Ejemplo de una arquitectura de capas (layers) https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html