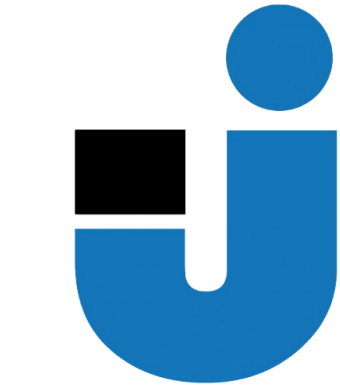


Universidad Nacional
ARTURO JAURETCHE

Metodología de la Programación II

Trabajo Práctico Final

Monasterio Salvatori Luparello Cantero

Noviembre del 2019

Índice

1. Introducción

En el siguiente informe se detalla lo realizado como parte del Trabajo Práctico Final de la materia **Metodología de la Programación II** para la **Comisión nº 2**. El presente trabajo se basa en la planificación y modelización de un negocio dedicado al alquiler de canchas de fútbol.

Asimismo el presente trabajo vendrá acompañado para una mayor comprensión de los siguientes ítems:

- Diagrama de Clases con sus correspondientes detalles y funcionalidades.
- Especificación de la Metodología Ágil utilizada.
- Especificación de los Patrones de Diseño que pueden ser utilizados.
- Código refactorizado.
- Soportes teóricos en los que se trabajó.

2. Alcance del Negocio

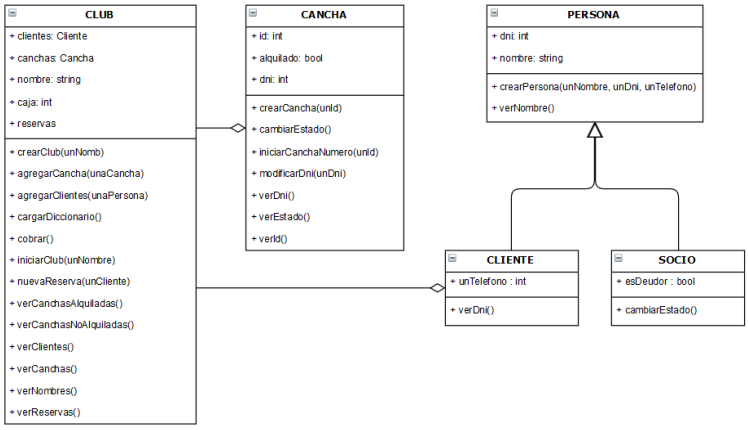
En un Club llamado "Tercer Tiempo" donde se alquilan 5 canchas para los Clientes. Todas ellas con el valor de $200 cualquiera sea el horario. Sólo es posible operar con los socios que ya estén inscriptos en el club, en caso que no se cuente con la inscripción, es posible realizarla al comienzo del proceso.

También se tiene un módulo de gestión de Datos el cual permite entre otras cosas, realizar lo siguiente:

- Generar un reporte de las personas que tienen alquilada una cancha.
- Generar un reporte de las canchas que **no** se encuentran alquiladas.
- Generar un reporte de las canchas que **se encuentran** alquiladas.
- Generar un reporte de los socios del Club en un momento determinado.

3. Diagrama de Clases

En el siguiente apartado se muestra mediante un esquema gráfico, la distribución de clases que se tuvieron para modelizar el negocio:

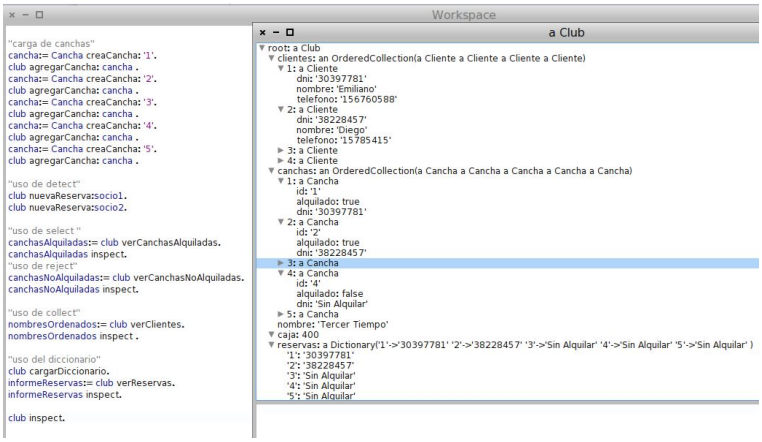


4. Funcionamiento

Para el modelado del negocio, se pensaron varias estructuras que permitan obtener determinados datos de forma sencilla, siendo los siguientes:

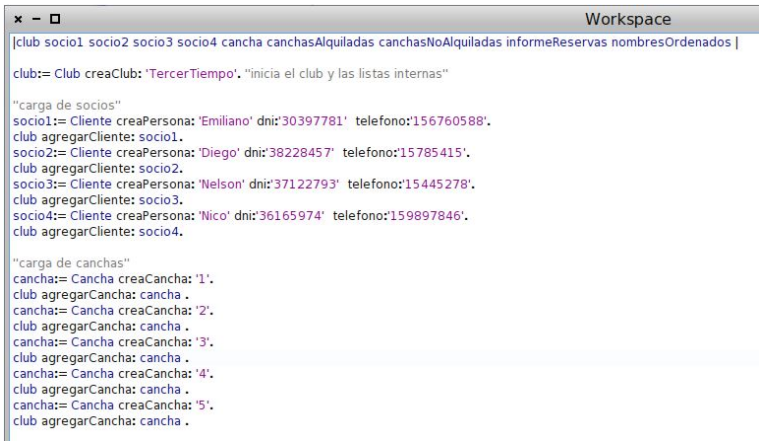
- **Detect**: permite obtener el primer objeto que cumple con la condición suministrada.
- **Select**: permite seleccionar varios objetos que cumplen con la condición ofrecida.
- **Reject**: permite seleccionar aquellos objetos que NO cumplen con la condición suministrada.
- **Collect**: permite generar una colección y ordenarla según el criterio dado por el programador.
- **Diccionario**: Un diccionario es un conjunto de definiciones que contiene las características lógicas y puntuales de los datos que se van a utilizar en el sistema que se programa.

Se puede visualizar a continuación un pantallazo general de lo obtenido luego de la carga de datos al programa:



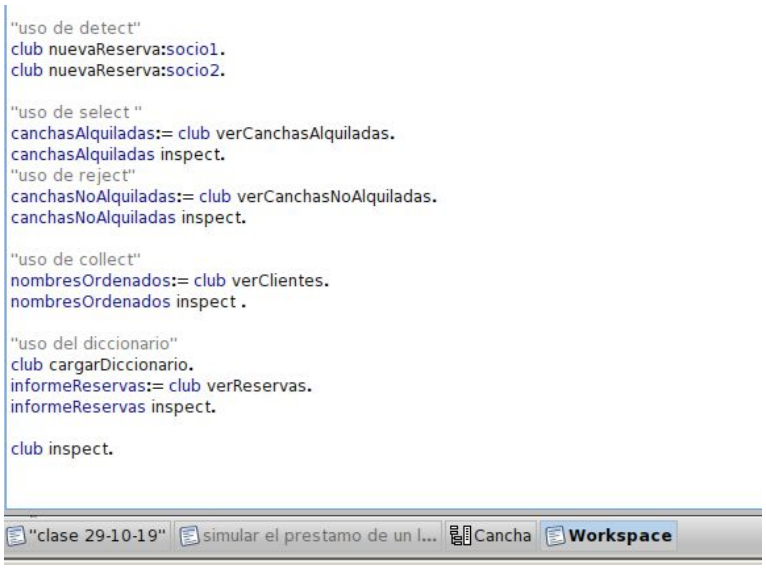
4.1. Set de datos utilizados

A continuación se visualiza el set de datos cargados para la exposición del Trabajo Práctico:



4.2. Mensajes empleados

Se visualizan los distintos mensajes que entiende la clase Club y las cláusulas utilizadas para cada uno:



4.3. Detect

Para la implementación de este procedimiento, se utilizó en la definición del método **nuevaReserva** el cual permite generar una nueva reserva consultando las canchas que NO se encuentran alquiladas, es decir, cuyo estado Alquilado sea igual a nil.

4.4. Select

Para la implementación de la cláusula *Select* se hizo uso de la misma en el método **verCanchasAlquiladas**, el cual permite obtener un reporte de las canchas que se encuentran alquiladas por los socios en un momento determinado.

4.5. Reject

Para la implementación de la cláusula *Reject* se hizo uso de la misma en el método **verCanchasNoAlquiladas**, el cual permite obtener un reporte de las canchas que NO se encuentran alquiladas por los socios en un momento determinado.

4.6. Collect

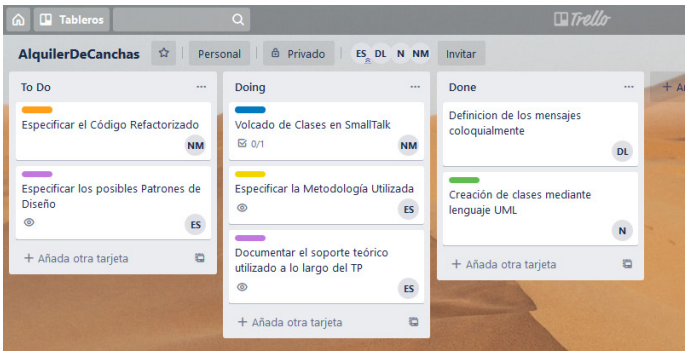
La cláusula *Collect* se implementó en el método **verClientes**, el cual permite generar una colección de los Clientes que tiene el Club, de forma ordenada alfabéticamente.

4.7. Diccionario

El sistema permite la utilización del método **verReservas** para saber a qué cliente le corresponde qué cancha. La misma permite obtener un reporte obteniendo la cancha alquilada y por quién se encuentra alquilada.

5. Metodologías Ágiles empleadas

Se decidió por implementar *Kanban* como soporte metodológico, ya que mediante la aplicación *Trello* [1] posibilitó trabajar de forma remota, a los cuatro integrantes del equipo. Como se puede ver a continuación se cargaron las siguientes tareas a realizar por el equipo:



6. División de tareas

A lo largo de todo el proyecto, se utilizaron distintas tarjetas en un pizarrón virtual para poder organizar entre los cuatro integrantes el trabajo a llevar a cabo. Es por eso que la distribución de las tarjetas y su cometido son los siguientes:

- **Definición de los mensajes coloquialmente**: Se utilizó el lenguaje coloquial para poder abordar de forma más sencilla las problemáticas que traían consigo cada una de las clases y el modelizado del negocio del Club.
- **Creación de Clases en Lenguaje UML**: Luego de haber definido los mensajes y las clases que intervendrían en el resultado, se pasó a crear las clases en lenguaje UML; para el cual fue utilizada la aplicación *Draw.io* [2]
- **Especificar la Metodología Utilizada**: Se requirió utilizar alguna de las Metodologías Ágiles vistas en a lo largo de la cursada. Dado el soporte de la herramienta *Trello* y sus características, se decidió implementar una pizarra al estilo Kanban; asimismo se añadió una especie de *Sprint backlog* para que cada integrante pudisese visualizar de forma gráfica la evolución de su trabajo.
- **Volcado de Clases en SmallTalk**: Luego de tener las clases y los diagramas que modelizarían el negocio, se llevó a cabo la codificación de las mismas en lenguaje *SmallTalk* [3], con asistencia del IDE denominado *Pharo* [4]

[1] Para más información, visitar: https://trello.com/
[2] Para más información, visitar: https://www.draw.io/
[3] https://en.wikipedia.org/wiki/Smalltalk
[4] Para más información, visitar: https://pharo.org/

- **Documentar Soporte Teórico**: Con la ayuda de la herramienta *Trello* se pudo dar igual seguimiento a todas las tareas realizadas por cada uno de los integrantes del equipo como también al soporte teórico implementado en el Trabajo Práctico.
- **Especificación del Código Refactorizado**: En muchas ocasiones se tuvo que volver sobre los pasos con respecto a las definiciones de los métodos de acceso a los datos de clase. Estos métodos redefinidos, ya sea por estar por fuera del alcance del método o por complejidades aparejadas a la forma de obtener un comportamiento, se refactorizaron para poder mejorar la calidad del código.
- **Especificación de Patrones de Diseño**: Si bien el proyecto es bastante pequeño como para poder aplicar Patrones de diseño, se pensaron algunos como para poder en un futuro expendir las funcionalidades del negocio. Se documentaron los posibles Patrones de Diseño a implementar en un futuro.

7. Soporte teórico implementado

Como se mencionó anteriormente, la Metodología Ágil escogida por el equipo fue *Kanban*. La misma se implementó teniendo en cuenta las siguientes características:

- **Comenzar con lo actual**: El método Kanban se inicia con las funciones y procesos que ya se tienen y estimula cambios continuos, incrementales y evolutivos al sistema. Es por ello que se partió de un bosquejo simple para esbozar la problemática a modelar para luego irle agregando complejidad a medida que se veía necesario.
- **Aplicar cambios incrementales**: Todos deben estar de acuerdo en que la manera de hacer mejoras en el sistema es el cambio continuo, gradual y evolutivo. Los cambios fuertes pueden parecer más eficaces, pero fracasan más debido a la resistencia y el miedo en la organización. El equipo siempre estuvo de acuerdo con los cambios propuestos por los distintos integrantes, para luego consultarlo a la docente.
- **Respetar lo establecido**: Para facilitar el cambio futuro conviene respetar los roles, responsabilidades y cargos actuales, eliminando los temores iniciales. Esto permite obtener un mayor apoyo a la iniciativa Kanban.
- **Liderazgo en todos los niveles**: Kanban promueve acciones de liderazgo desde las personas de bajo rango hasta los gerentes. Esto fue tenido en cuenta a la hora de decidir qué modificar sin tener en cuenta la tarea asignada por la persona.

El equipo se basó por sobre todas las cosas en **Visualizar el trabajo**. Kanban se base en el desarrollo incremental, dividiendo el trabajo en partes. Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada tarea, y que se representa en pizarras llenas de post-it.

Normalmente cada una de las partes del trabajo se escribe en un post-it y se pega en un pizarron. Los post-it pueden tener información variada, aunque en particular deberían tener la estimación de la duración de la tarea. La tarea de escribir la ocupación de cada integrante fue llevada a cabo mediante la herramienta Trello.

Así como la metodología Kanban no prescribe roles, ya que tener un papel asignado y las tareas asociadas a dicho papel crean una identidad en el individuo, también presentó la posibilidad de que todo el equipo desarrolle todo.

Se puede también dar cuenta de que se implementó lo denominado como *Spring Backlog*, el cual es la lista de tareas que el equipo elabora como plan para completar los requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de incremento de producto preparado para ser entregado.

En este caso, se puede tomar el Cliente como el docente que estará evaluando la implementación, la documentación realizada, como así también los métodos para llevar a cabo el propósito del Trabajo Práctico.

8. Código Refactorizado

En la etapa del volcado de código, se mejoraron los nombres de muchas variables para hacerlas lo más descriptivas posibles.

Asimismo se mejoró el código del método **nuevaReserva** que implementa en la clase Club. Este fue refactorizado debido a que en caso de no encontrarse un objeto que cumpla con las condiciones suministradas, no devolvía nada, por lo que se implementó la cláusula *ifNone(ñil)*, la cual en caso de que no se cumpla con la condición suministrada, devuelve nulo.

9. Framework

Ya que la implementación se realizó en el lenguaje *SmallTalk*, se indagó en qué tipo de Frameworks sería el más conveniente para poder adaptar el modelo de negocio a las nuevas tecnologías, por lo que se pensó en *Seaside*, el cual proporciona un conjunto de abstracciones en capas sobre HTTP y HTML que le permiten crear aplicaciones web altamente interactivas de forma rápida, reutilizable y mantenible. [5]

10. Escalabilidad

Como parte del proyecto, también se pensó en dejar abierto el código para futuras funcionalidades, permitiendo la escalabilidad al implementar otras clases como ser la de **Socio**, la cual tiene como dato miembro la variable *esDeudor*, que permite saber si el mismo está al día con la cuota del club o no.

En caso de que no lo esté, también se pensó que se puede aplicar una multa, pagando más dinero en el alquiler de las canchas.

[5] https://www.bestwebframeworks.com/web-framework-review/smalltalk/106/seaside/

11. Patrones de Diseño

Siguiendo la idea que hay detrás de todo Patrón de Diseño, el cual es *una forma reutilizable de resolver un problema común*, y teniendo en metente lo mencionado en el apartado anterior sobre la Escalabilidad, se pensó en que se puede aplicar al modelado del negocio de las canchas el patrón **Decorator**, el cual permitiría añadir funcionalidad extra a un objeto (en este caso las canchas y los descuentos para socios NO deudores) sin modificar el comportamiento del resto de objetos del mismo tipo (es decir, la creación que se está utilizando hasta el momento).