**Planificación**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Descripción** | **Esfuerzo(hrs)** | **Predecesoras** |
| A | Función para cargar los socios al sistema | 1 | - |
| B | Cargar disponibilidad de canchas por horas dada una fecha | 2 | - |
| C | Realizar Vistas | 5 | - |
| D | Función tarifa | 3 | - |
| E | Agregar reserva | 5 | A-B-C-D |
| F | Cargar reservas de clientes | 3 | E |
| G | Validar asistencia | 3 | F |
| H | Aplicar multa por inasistencia | 3 | F |
| I | Cancelar reserva | 3 | F |
| J | Aplicar multa por cancelar reserva | 3 | I |
| k | Cargar multas de clientes | 3 | H-J |
| L | Validar pago de multas | 1 | K |

**Camino crítico:** C-E-I-J-K-L

**Total:** 25 horas

**Patrones de diseño**

***Singleton***

* **Sistema:** se decidió hacer un singleton para que funcione como una clase estática, que sólo proporciona métodos con un fin concreto.
* Los distintos controladores deben ser únicos para evitar conflictos. Estas a su vez se logran sincronizar datos con la clase sistema.

***MVC***

Diseño basado en la arquitectura lógica MVC, debido a que se puede determinar cuáles serán las vistas, con que clases interactuarán y se desmiembra el control de las clases del modelo.

**Funcionalidades Faltantes**

* La aplicación no provee la multa por inasistencia
* Los clientes y canchas fueron creados manualmente dentro de la aplicación (en la clase Sistema). El número de canchas fue asignado en la clase CommandNames
* Las tarifas se encuentras asignadas en la clase CommandNames