

Ingeniería de Software

Proyecto: La Mosca

Docentes



Martín
Miceli

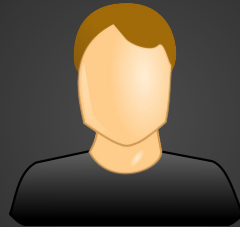


Martín
Bustos

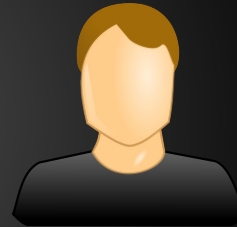
Equipo



Agustín
Carranza



Bruno
García



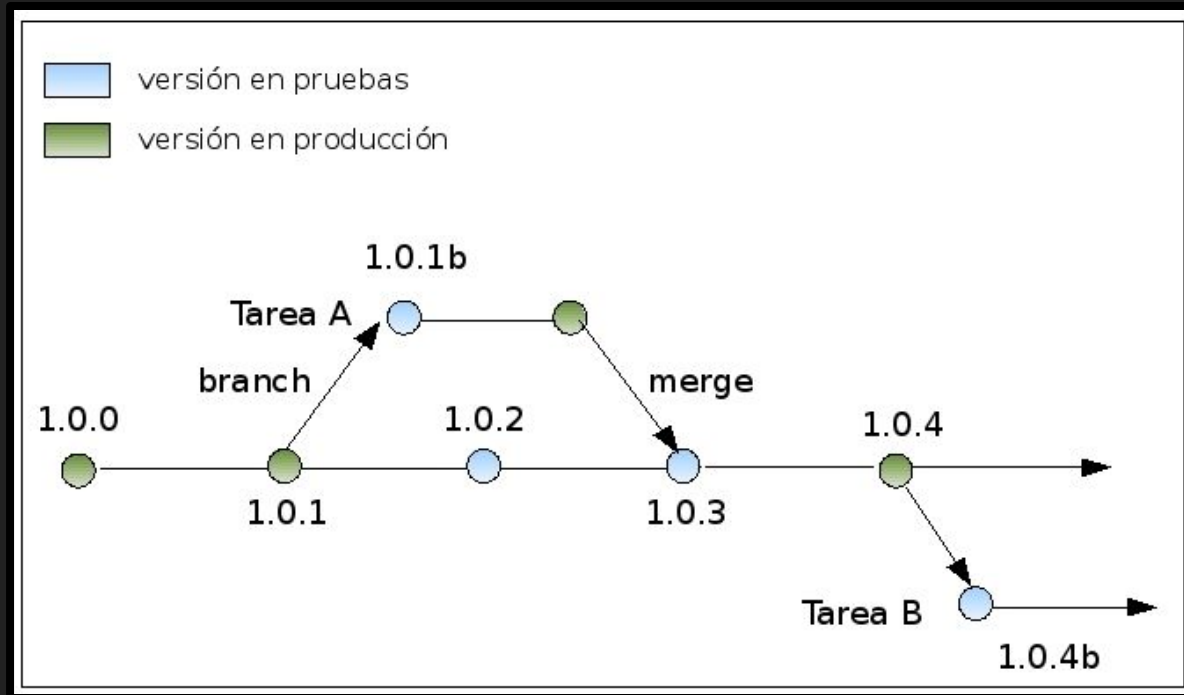
Emanuel
Echazú

Configuration Management

Configuration Management

Lenguaje de programación	Java
Entorno de desarrollo	IntelliJ IDEA Ultimate Edition
Sistema de control de versiones	Github
Sistema de seguimiento de errores	Github Issues
Herramienta de integración continua	Travis CI
Herramienta de automatización	Maven

Configuration Management



Requerimientos

Requerimientos

Funcionales

- El juego se desarrolla en un tablero gráfico
- Modalidad de juego usuario vs PC
- Se reparte 5 cartas a cada jugador
- Tras jugar 5 bazas, se muestran los puntos obtenidos
- El ganador es el usuario con mayor puntaje

Requerimientos

No Funcionales

- Funcionamiento en JRE (Java Runtime Environment)
- Control por medio de mouse y teclado
- Respuesta del sistema al usuario en menos de 2 segundos

Diseño de la Arquitectura

Patrón Model View Controller

MVC es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

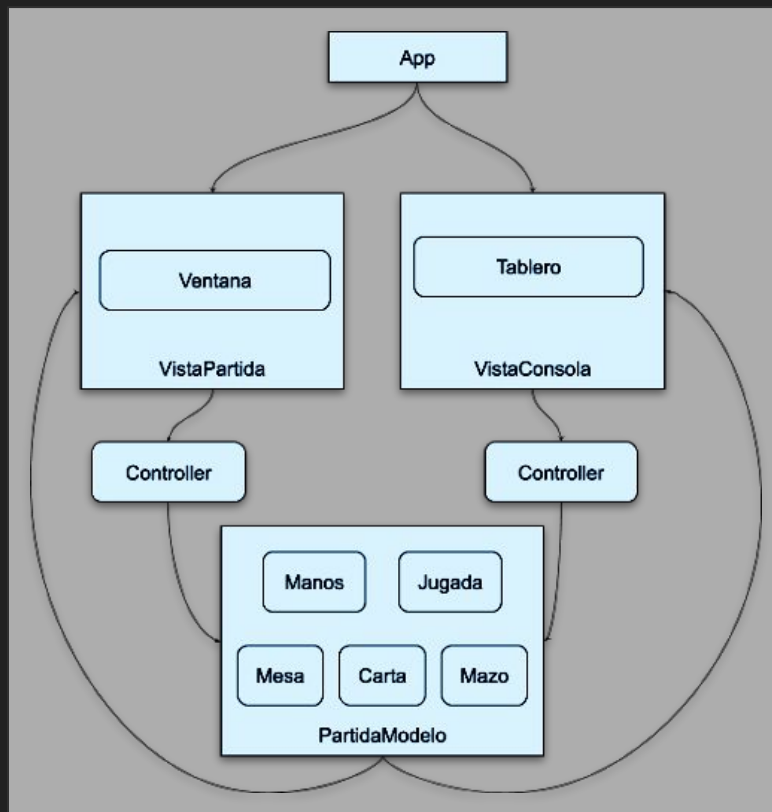
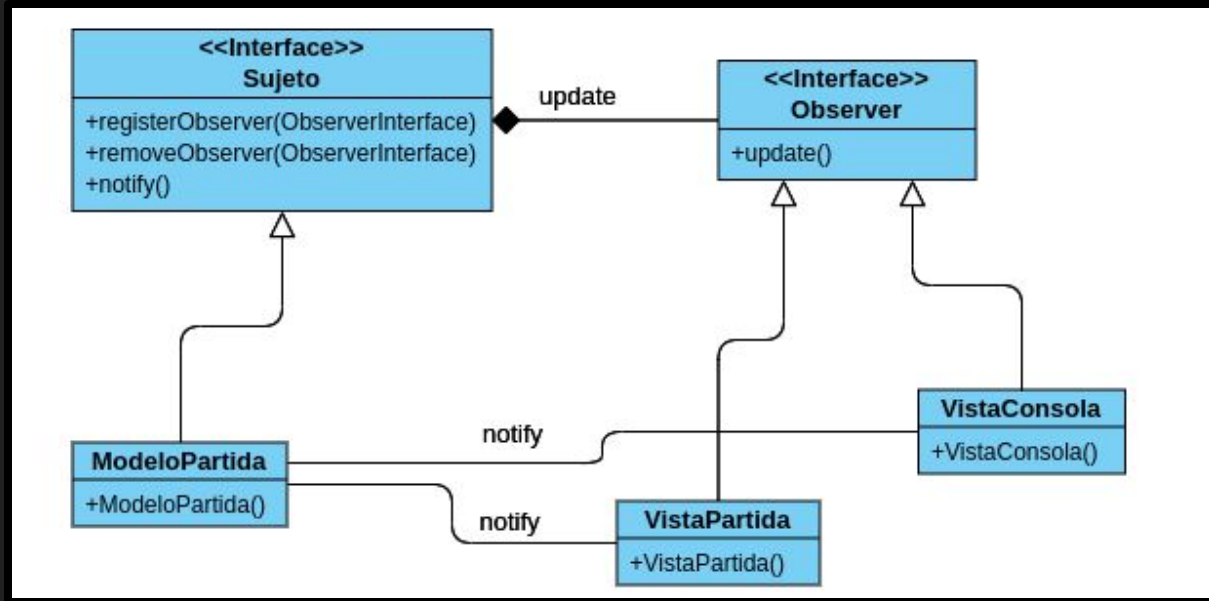


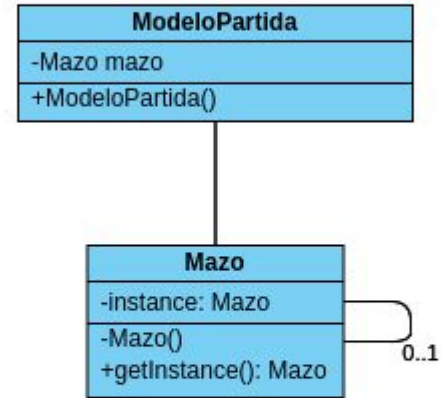
Gráfico de Arquitectura - Patrón MVC

Diseño e Implementación

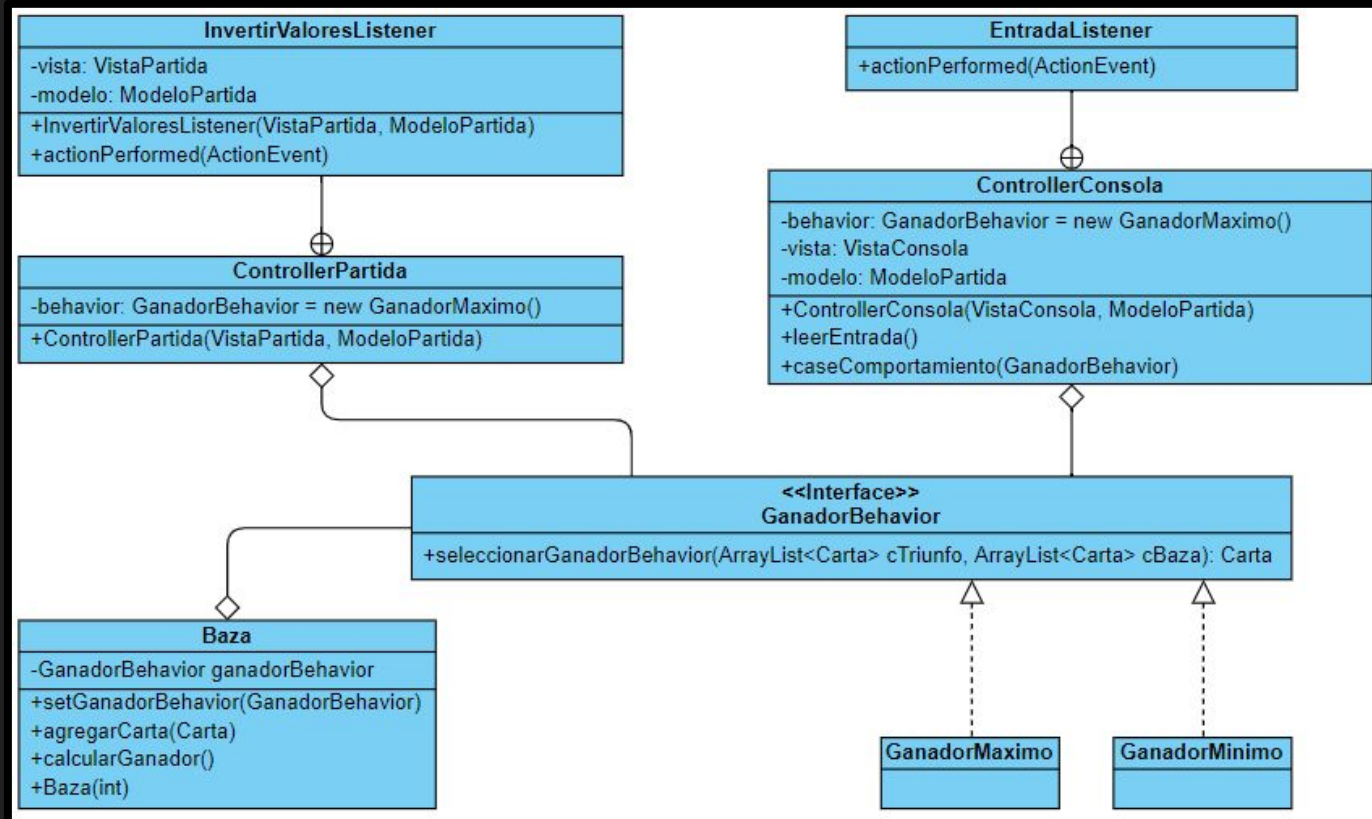
Patrón Observer



Patrón Singleton



Patrón Strategy



Diagramas de Clase

Diagrama de Clase del Modelo

com.model

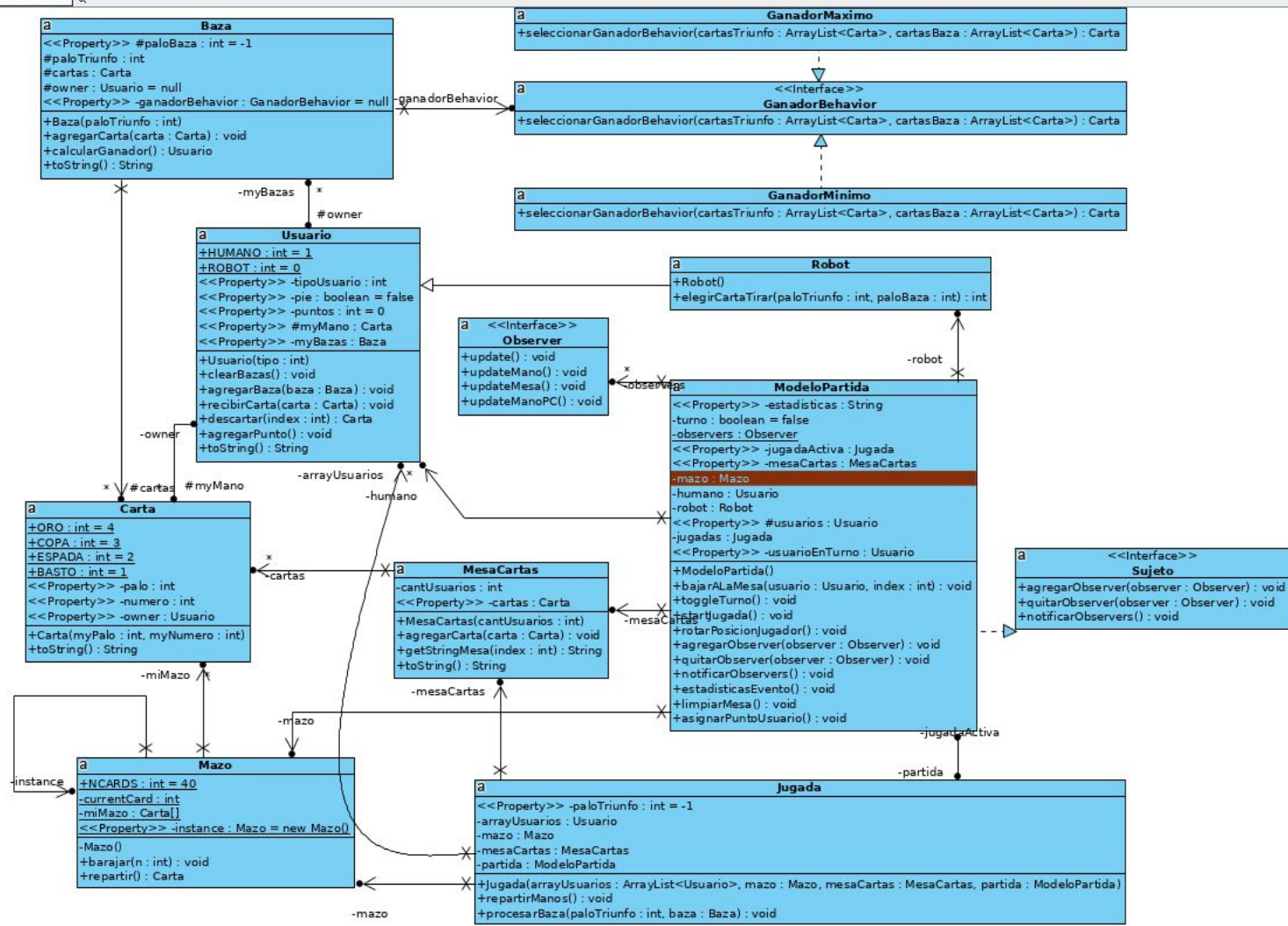


Diagrama de Clase de las Vistas

com.view

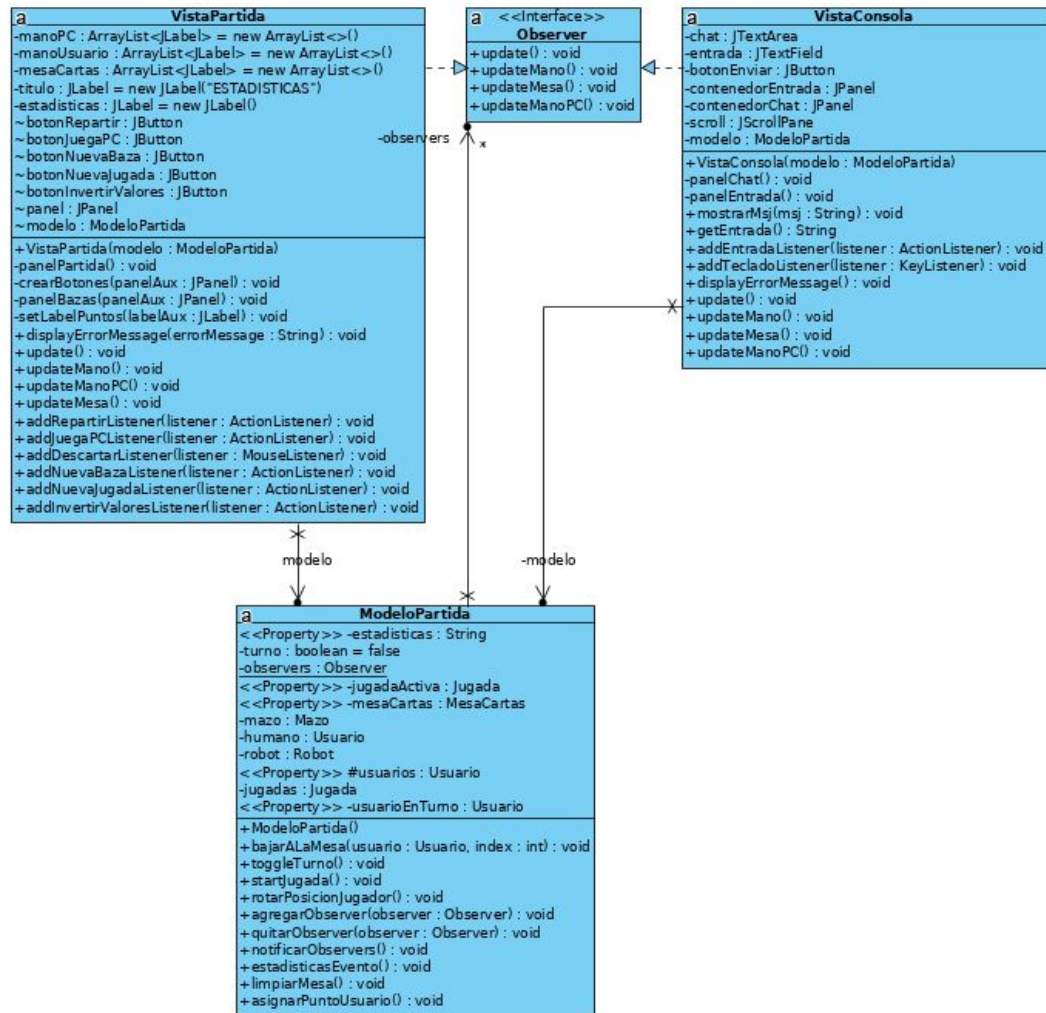


Diagrama de Clase de los Controladores

com.controller

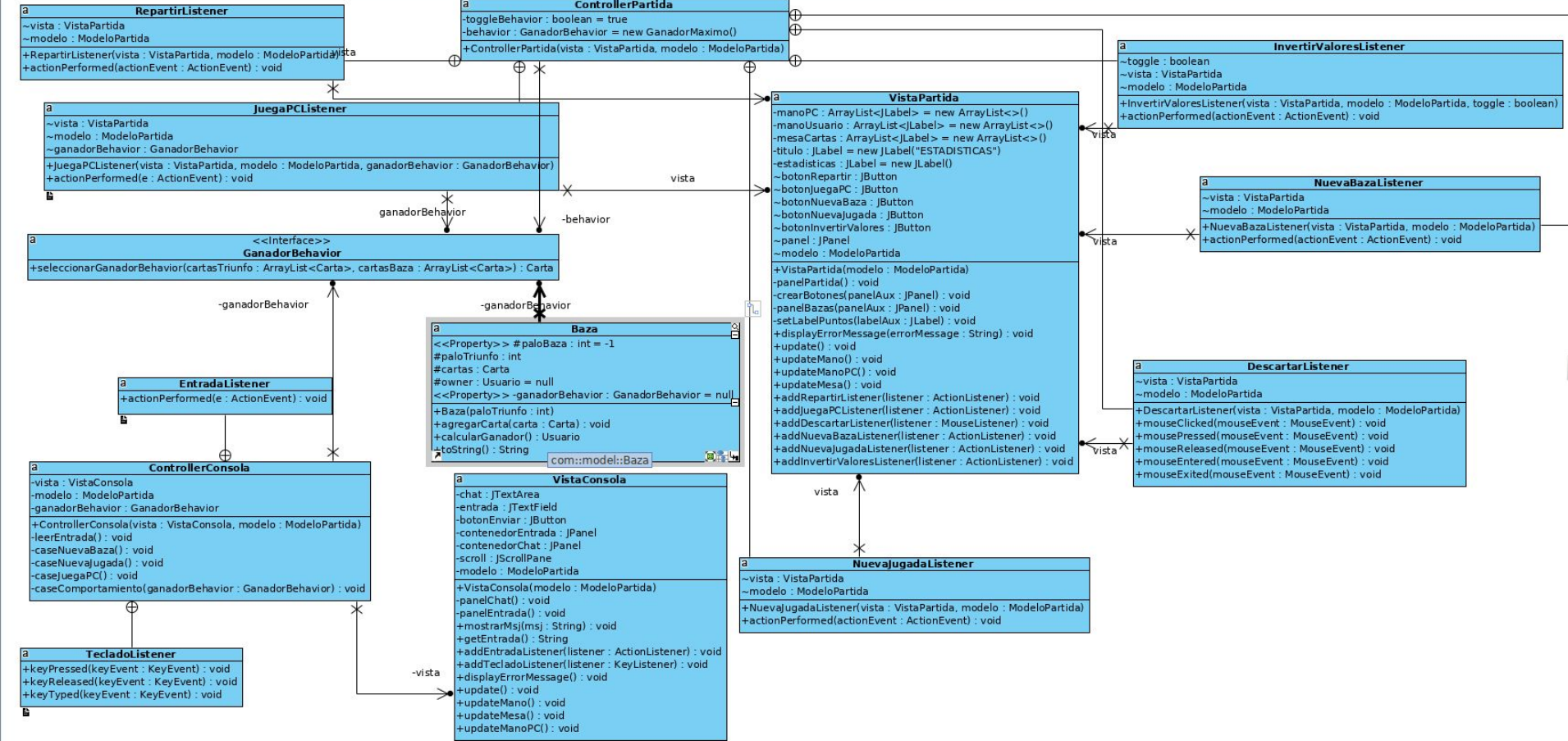


Diagrama de Despliegue

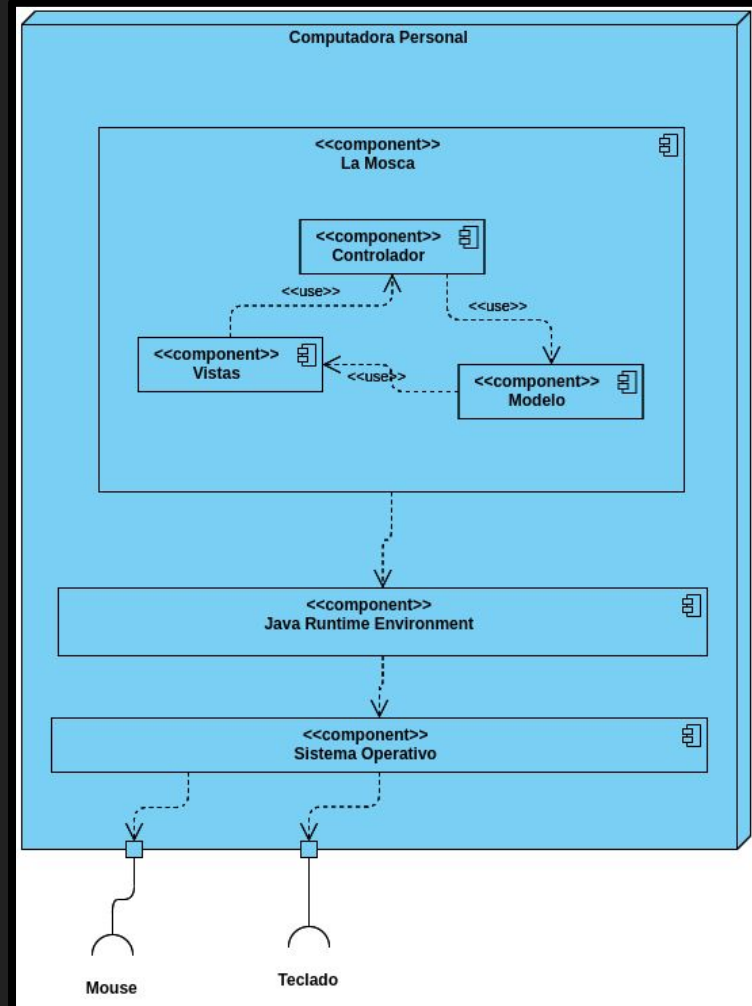
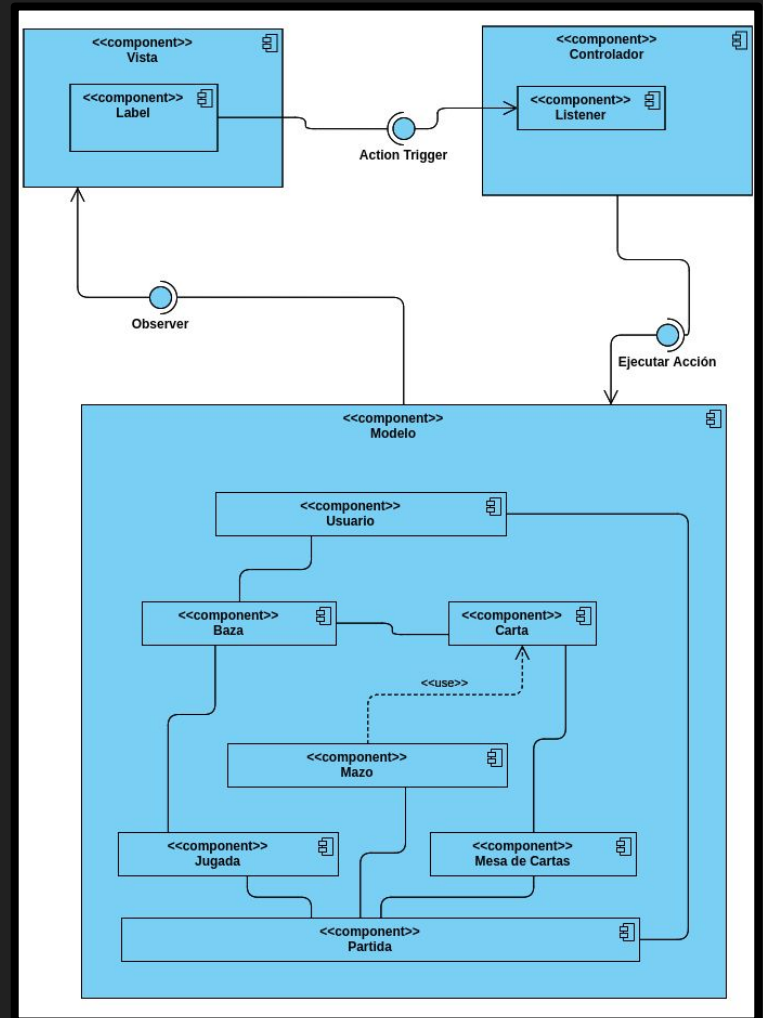
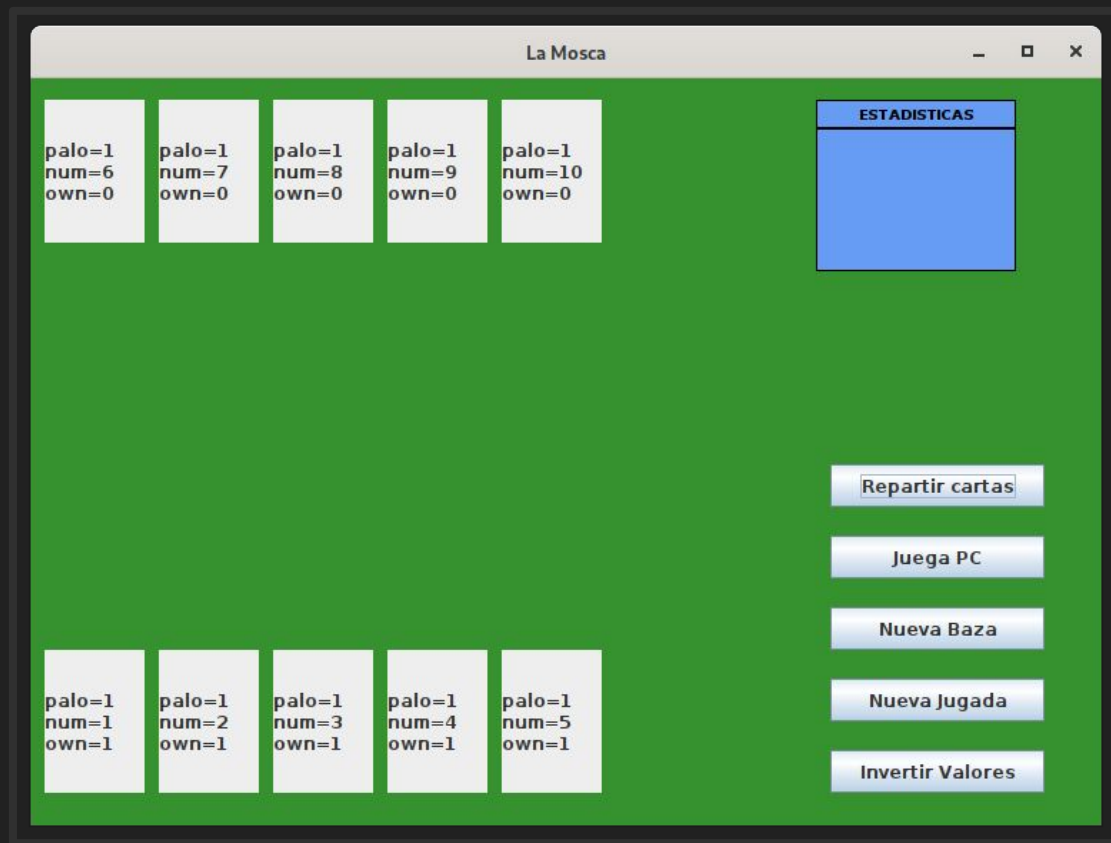


Diagrama de Componentes

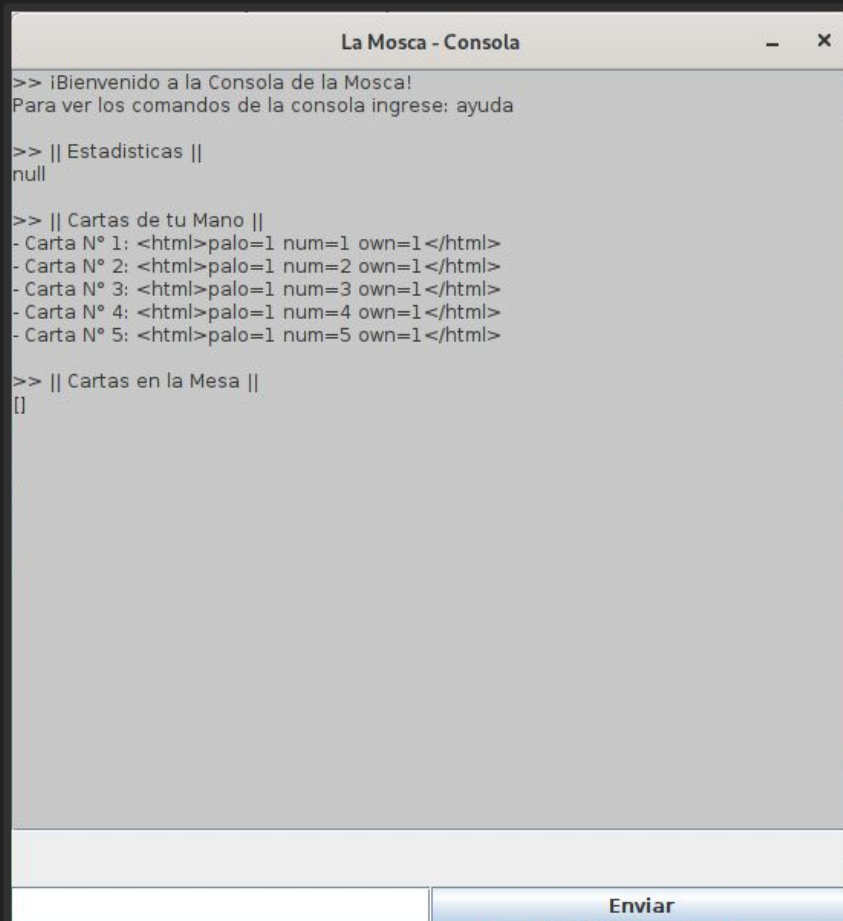


Vistas

Vista Gráfica



Vista Consola



The screenshot shows a web browser window with a console titled "La Mosca - Consola". The console contains the following text:

```
>> ¡Bienvenido a la Consola de la Mosca!  
Para ver los comandos de la consola ingrese: ayuda  
  
>> || Estadísticas ||  
null  
  
>> || Cartas de tu Mano ||  
- Carta N° 1: <html>palo=1 num=1 own=1</html>  
- Carta N° 2: <html>palo=1 num=2 own=1</html>  
- Carta N° 3: <html>palo=1 num=3 own=1</html>  
- Carta N° 4: <html>palo=1 num=4 own=1</html>  
- Carta N° 5: <html>palo=1 num=5 own=1</html>  
  
>> || Cartas en la Mesa ||  
[]
```

At the bottom of the console, there is a text input field and a button labeled "Enviar".

Testing

Tests Automáticos

Results :

Tests run: 14, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO] BUILD SUCCESS

[INFO]

[INFO] Total time: 2.983 s

[INFO] Finished at: 2020-06-24T00:22:08-03:00

[INFO]

Lecciones Aprendidas

Lecciones Aprendidas

Implementar los patrones de diseño strategy y observer no implica haber utilizado el patrón de arquitectura MVC.

Se puede implementar el patrón de arquitectura MVC y no utilizar los patrones de diseño antes mencionados.

Es muy productivo utilizar patrones que desacoplan funcionalidades ya que permite el desarrollo paralelo, la división de tareas y la optimización de los tiempos.

A la hora de estimar tiempos de desarrollo debe tenerse en cuenta las fortalezas y debilidades de los miembros del equipo.

Lecciones Aprendidas

Para poder desarrollar un proyecto profesional siguiendo los pasos correctos, en el orden que corresponde y utilizando metodologías acordes, es requisito fundamental tener experiencia en este tipo de trabajo.

Si el equipo tuviera que realizar un nuevo proyecto luego de haber culminado las actividades propuestas, seguramente transitaría el desarrollo de todas las etapas de forma más fluida, pudiendo comunicarse con terminología y herramientas ya conocidas, y poniendo foco en el proyecto propiamente dicho.

Errores Cometidos

Errores Cometidos

Al momento de la elección del tema del proyecto, el equipo no pudo dimensionar la dificultad y el tiempo que insumiría la implementación de las reglas del juego.

De haber elegido una metodología de trabajo con énfasis en el testing, se podrían haber desarrollado un mayor número de pruebas automáticas.

Para aprovechar todas las virtudes que significa implementar el patrón de arquitectura MVC, el equipo podría haber diseñado dos modelos (por ejemplo, agregar otro juego de cartas con reglas distintas). Entonces el controlador, usando el patrón de diseño strategy, permitiría elegir en tiempo de ejecución un modelo u otro.

Errores Cometidos

En la etapa de gestión de las configuraciones, se optó en un comienzo por la herramienta Jenkins para la integración continua. No se tuvo en cuenta que ésta requiere ser instalada en un servidor dedicado. El equipo no contaba con uno e implementarlo excedía el alcance del proyecto. Finalmente se optó por la herramienta Travis que corre en la nube.

FIN