Arquitectura del Sistema

Ingeniería del Software

Nombre del grupo: The Moth

Nombre del proyecto: La Mosca

Nombre del documento: Plan de Gestión de las Configuraciones

Integrantes: - Agustín Carranza
- Bruno García
- Emanuel Echazú

Fecha: 23/06/2020

Versión del documento: 1.0.0

Versiones del Documento

Versión	Fecha	Resumen del Cambio	Autores
0.1.0	20/05/2020	Creación del documento.	Emanuel Echazú Bruno García Agustín Carranza
0.2.0	16/06/2020	Agregación de diagramas de arquitectura.	Agustín Carranza
0.3.0	19/06/2020	Agregación de diagramas de arquitectura y creación de pruebas de integración.	Emanuel Echazú Bruno García Agustín Carranza
1.0.0	23/06/2020	Corrección de las pruebas de integración. Se realizaron las correcciones en base a la devolución de Martín B.	Emanuel Echazú

Índice

Consigna

Se debe añadir al documento de requerimientos creado en el trabajo práctico anterior una sección que incluya un **diagrama de arquitectura preliminar** que permita asociar requerimientos y casos de usos con los sistemas, subsistemas y módulos identificados.

Por otro lado, se debe generar otro documento de **arquitectura** donde se presente:

- Un gráfico de arquitectura general para mostrar los componentes y sus relaciones con las interfaces externas.
- La explicación del **patrón de arquitectura** que fue usado y por qué, haciendo énfasis en cómo resuelve los requerimientos no funcionales.
- Diagramas UML de despliegue y de componentes.
- Opcionalmente se puede incluir un diagrama de contexto que incluya la relación de los sistemas y los subsistemas con las actividades del dominio del conocimiento de la aplicación.

Se deben definir los casos de prueba de integración que verifican la correcta interacción entre todos los componentes del sistema.

Diagrama de Arquitectura Preliminar

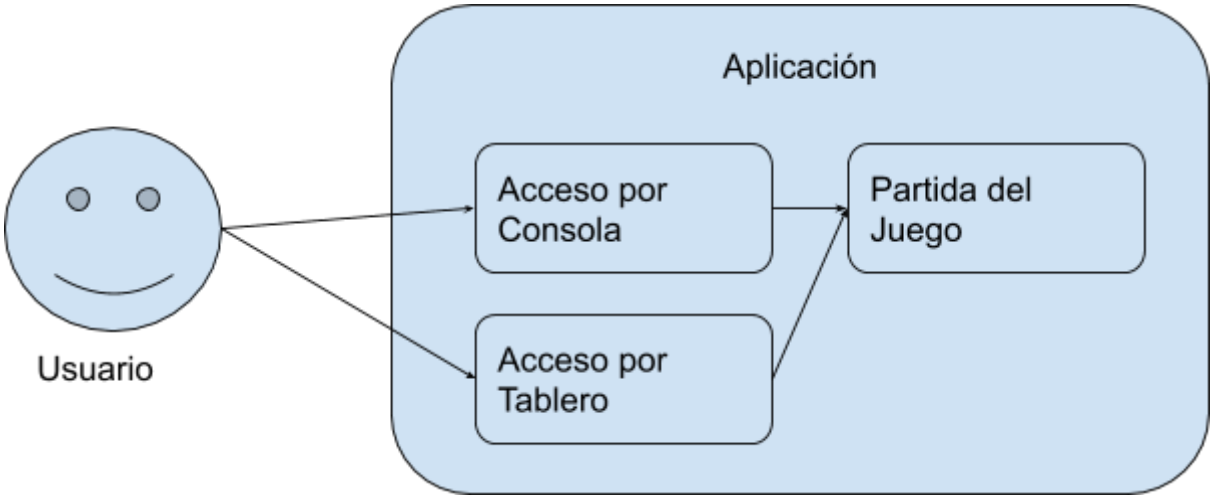
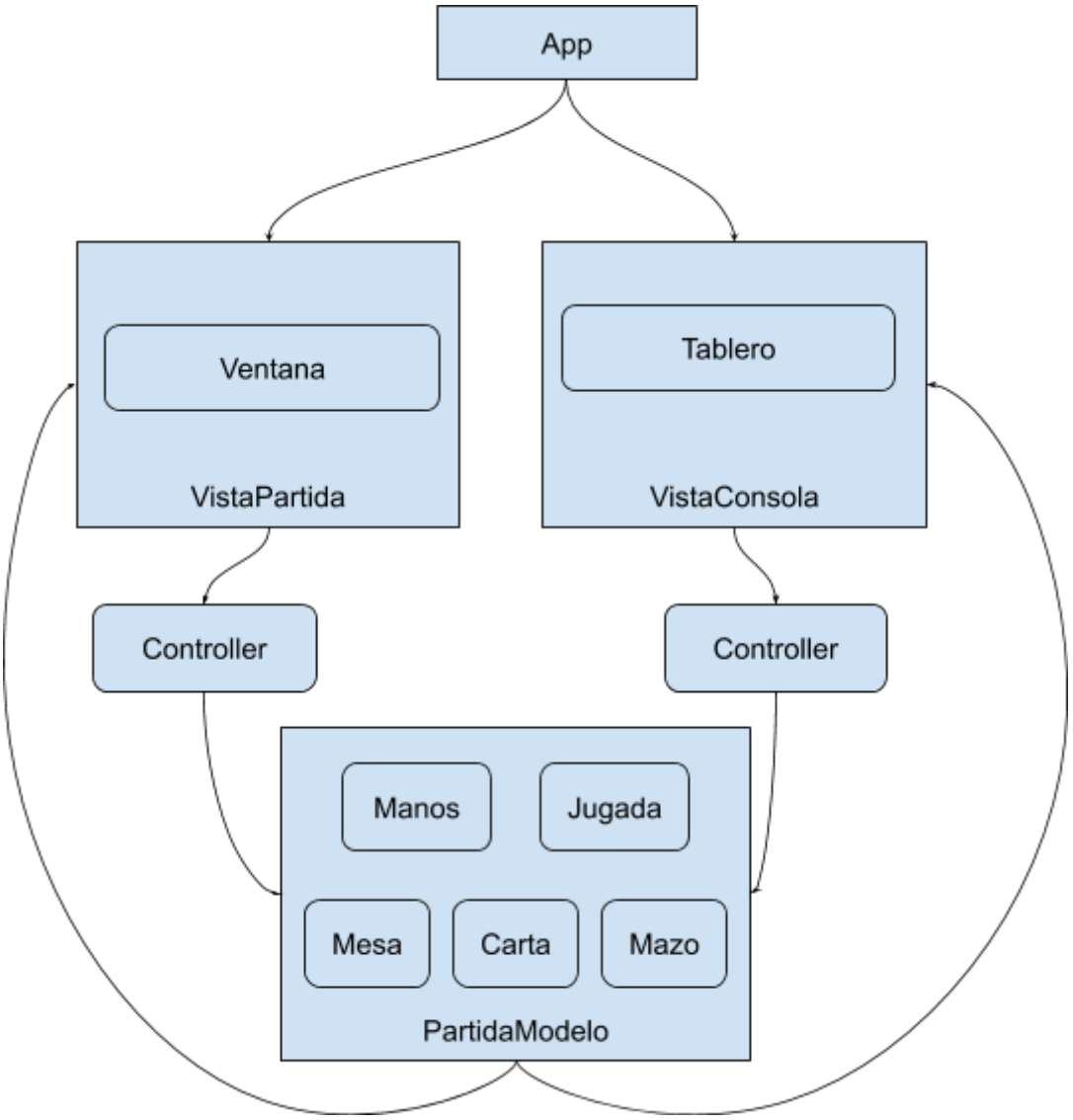


Gráfico de Arquitectura



Patrón de Arquitectura

Modelo-vista-controlador (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.

De manera genérica, los componentes de MVC se podrían definir como sigue:

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Se escogió este patrón debido a que es el más común a la hora de construir aplicaciones con interfaz gráfica. También favorece el trabajo en equipo debido a que los componentes están bien definidos, lo que permite la distribución de tareas.

Habida cuenta que el lenguaje elegido para el desarrollo es Java, los programadores cuentan con numerosas librerías que facilitan la implementación del modelo. Una de ellas es Swing que permite la interacción del usuario con una interface amigable usando teclado y mouse.

El código generado en Java permite ejecutarse en cualquier sistema operativo que haya implementado la JRE (Java runtime environment).

Diagramas UML

Diagrama de Despliegue

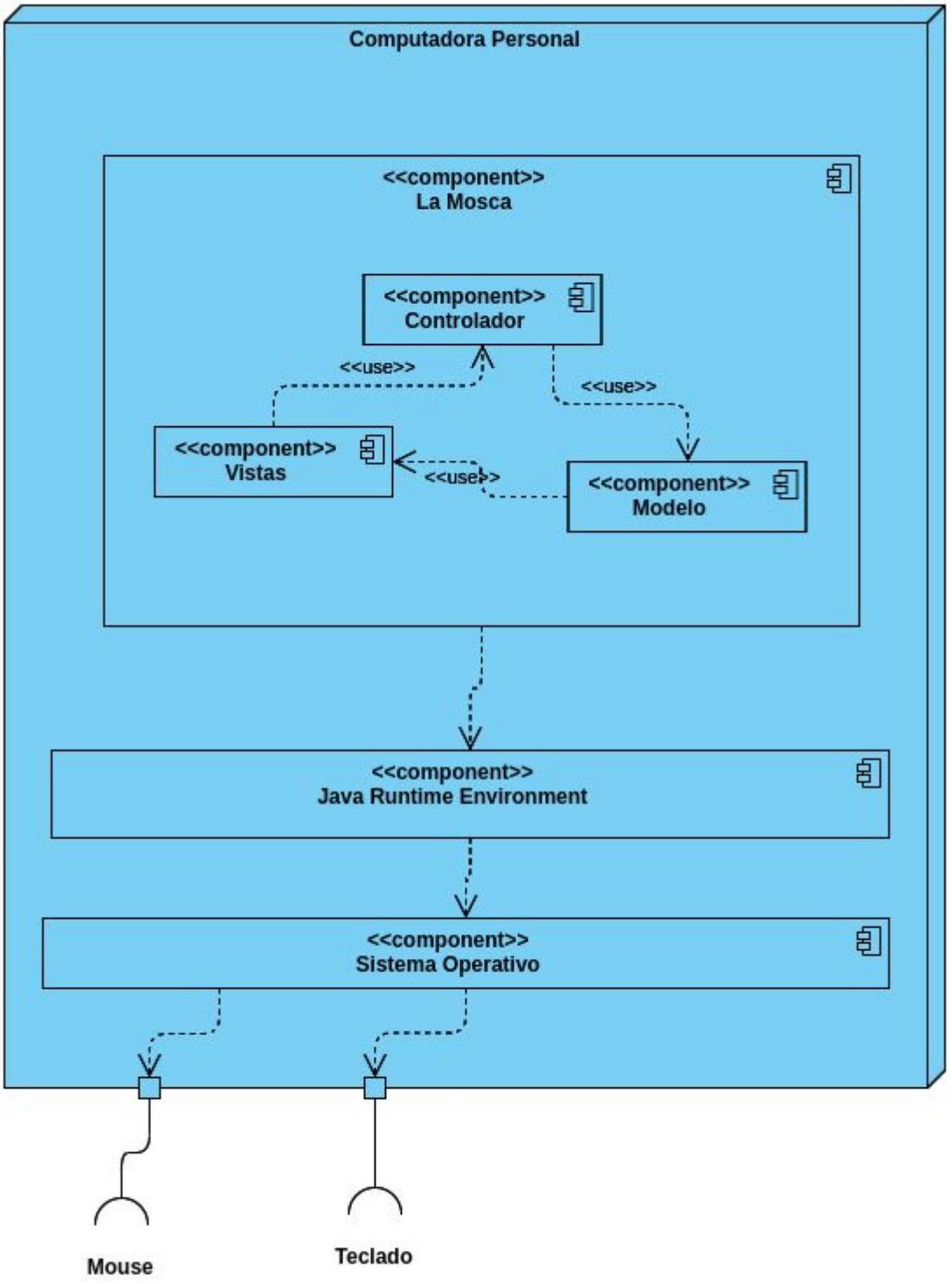
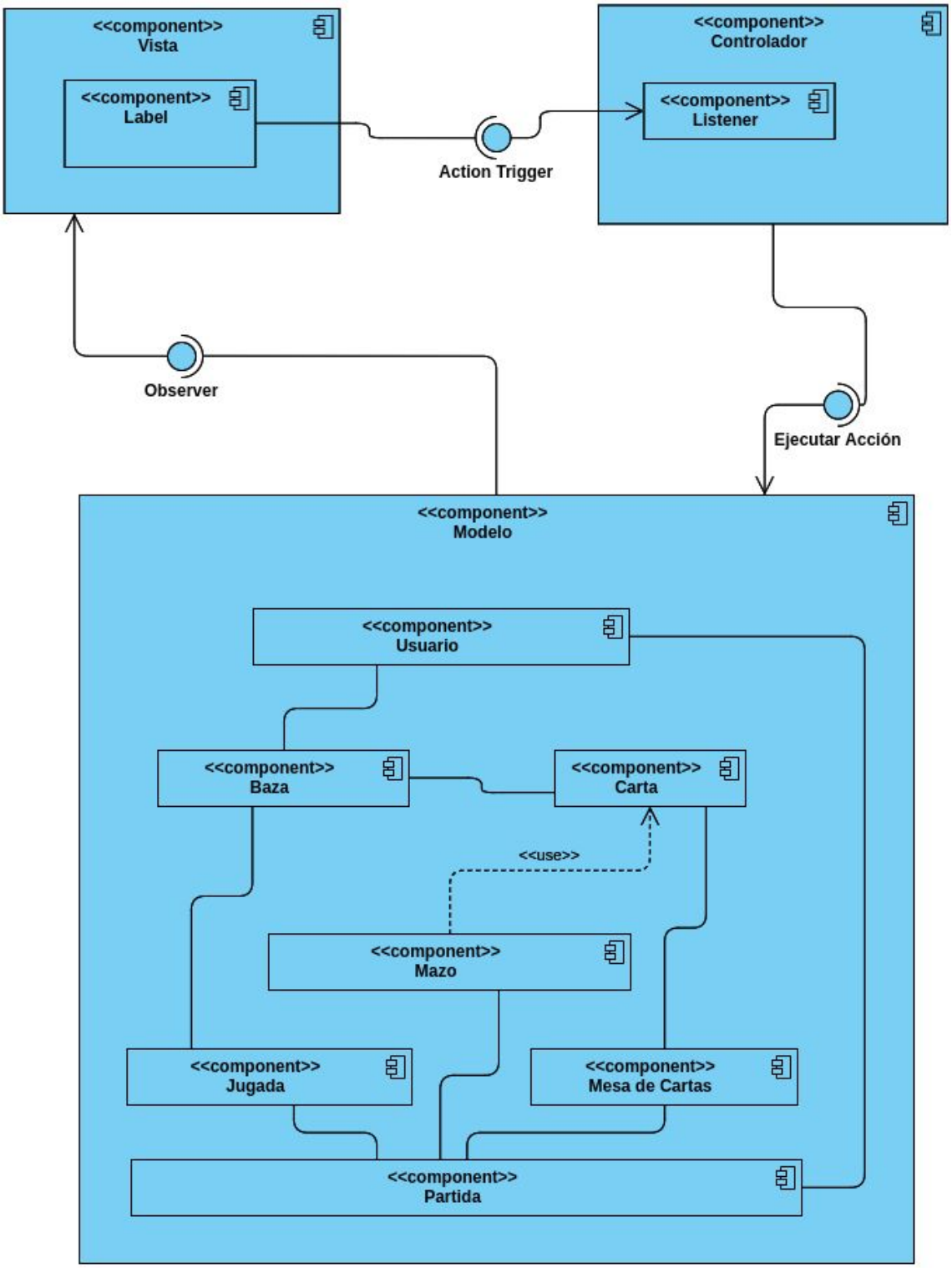


Diagrama de Componentes



Pruebas de integración.

El fin de estas pruebas es verificar la correcta interacción entre dos o más componentes del software actuando en conjunto. Este tipo de pruebas son dependientes del entorno en el que se ejecutan. Si fallan, puede ser porque el código esté bien, pero haya un cambio en el entorno.

Identificador	Propósito	Descripción	Resultado esperado
PI1	Verificar la jugada de descarte	Al momento de bajar una carta a la mesa se descarte de la mano del usuario y figure en la mesa de cartas	Las cartas se reacomodaron en la vista
PI2	Verificar el posicionamiento de los componentes	Comprobar que al empezar una jugada, los usuarios, el mazo y la mesa estén centradas	Los componentes figuran en las posiciones seteadas
PI3	Verificar la puntuación después de una jugada	Verificar la asignación de puntos al usuario ganador, en base al comportamiento de selección de carta ganadora en curso	El usuario ganador tenga un punto más
PI4	Verificar el orden asignado al mezclar el mazo	Cuando se repartan las cartas a cada jugador tiene que seguir el orden en que se acomodaron al barajar el mazo	El orden de cartas repartidas a los usuario es la misma que las mezcladas en el mazo al principio
PI5	Ver que la jugada se ejecute según las reglas establecidas	Verificar que el usuario ganador de una baza , dados los palos de baza y triunfo, y las cartas descartadas por los usuarios	El usuario ganador es el que tenía la carta más alta según los palos de baza y triunfo
PI6	Verificar que se crean las vistas con sus componentes	Probar que los componentes (botones y labels) se agregan a las vistas	La vista creada tiene sus componentes agregados según los valores establecidos
PI7	Verificar que no bajen a la mesa más cartas de las permitidas	Una vez realizada una jugada por parte de ambos jugadores, comprobar que las cartas en la mesa sean 2, que se agreguen a la baza del usuario ganador las cartas jugadas	La cantidad de cartas en la mesa son como máximo 2
PI8	Verificar el comportamiento de descarte	Cuando los jugadores realizan un descarte comprobar que el número de cartas que tienen en su mano disminuya en una	Los componentes cartas de mano del usuario que se descarte disminuya en 1