Documento de Arquitectura – PinturECI

David Enrique Pérez, Juan Pablo Espinosa, Nicolás Camacho Hurtado.

Arquitectura de Software (ARSW)

Escuela Colombiana de Ingeniería Julio Garavito

2022

1. Introducción y contexto:

La implementación del proyecto se basa en una adaptación del famoso juego online “pinturillo” en el que los jugadores dibujan para acertar una palabra y así ganar puntos. En la cual los jugadores podrán acceder desde diferentes dispositivos ingresar a la ronda de juego y interactuar en el juego de dos maneras, como dibujante, dibujando la palabra relacionada a la ECI, o bien como participante intentando adivinar la palabra a partir de la ilustración creada por el dibujante.

En este documento se explicará la arquitectura usada en el proyecto bajo vistas y modelos que permiten realizar un mejor entendimiento de este, como también se realizara una explicación de los requerimientos funcionales, no funcionales y atributos de calidad usados para el desarrollo del videojuego.

2. Objetivo:

Brindar una guía que permita plasmar de una manera integral las decisiones de arquitectura tomadas para la implementación del proyecto “PinturECI”, teniendo como base estricta modelos de arquitectura, requerimientos funcionales y no funcionales del sistema, atributos de calidad, y restricciones.

3. Documento de arquitectura

Historial de versiones:

El proyecto se encuentra asociado al sistema de control de cambios y versionamiento de GitHub para asegurar la trazabilidad de este, exactamente en el siguiente enlace:

https://github.com/DavidPZ666/pintureci

Software de administración del proyecto:

Se hace uso del software que permite implementar el ciclo de vida del desarrollo, para la metodología SCRUM, en la cual se especifican las historias de usuario del proyecto:

https://tree.taiga.io/project/juan-espinosac-pintureci/timeline

Software de análisis estático de código:

El proyecto se encuentra asociado al software SonarCloud que no permite analizar y cuantificar la calidad del código fuente, como también se encuentra asociado al software de integración continua CircleCI:

https://sonarcloud.io/summary/overall?id=DavidPZ666_pintureci

https://app.circleci.com/pipelines/github/DavidPZ666/pintureci?branch=master

Requerimientos Funcionales:

- Sistema de colaboración en tiempo real.

- Orientado a múltiples usuarios concurrentes. Esto implica que múltiples usuarios podrán interactuar con la plataforma al mismo tiempo en tareas completamente diferentes.

- El usuario que desee ingresar al aplicativo deberá digitar e ingresar su nickname y este no podrá ser igual a uno ya creado.

- El aplicativo debe contar con un mecanismo que permita iniciar la ronda y darle continuidad al juego.

- Una vez iniciada la ronda de juego, tanto el dibujante como el participante deberán tener una ventana tipo canva que permita realizar un dibujo con diferentes opciones de color o visualizar la ilustración del dibujante, todo esto dependiendo del rol del usuario (Dibujante o participante).

- Una vez iniciada la ronda de juego, todos los usuarios deberán tener un chat global que les permitirá comunicarse mediante cadenas de texto, como también en el caso de los participantes a adivinar la palabra que el dibujante se encuentra ilustrando.

Supuestos:

- Se infiere que el proyecto únicamente se desarrolla como aplicativo WEB que pueda ser usado únicamente con dispositivos que cuenten con un navegador web, bien sea un equipo de cómputo o un dispositivo móvil.

Restricciones:

- El sistema de las enfermeras debe estar soportado para los navegadores convencionales (Google Chrome, Mozilla Firefox, Microsoft Edge, etc…).
- La implementación del proyecto se debe realizar bajo el lenguaje Java, JavaScript.
- La mayoría de los componentes y despliegue deben ser alojados en la nube Azure.

Decisión Arquitectónica:

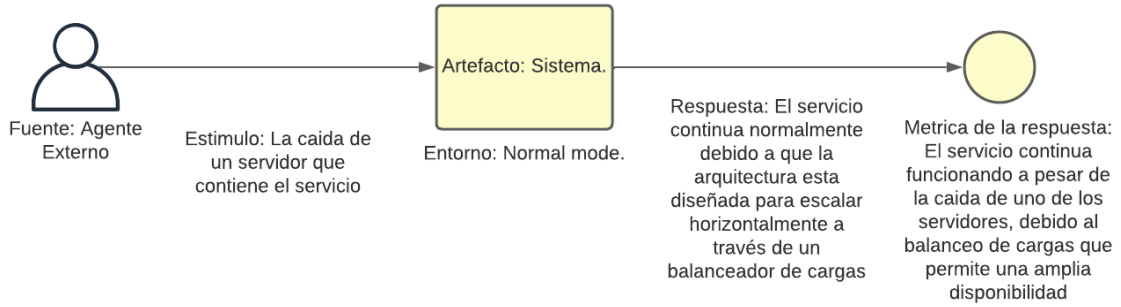
Para la arquitectura implementada en el backend del aplicativo se utiliza la arquitectura MVC (Modelo, Vista, Controlador) debido a que esta arquitectura permite una separación del Modelo y la Vista separando los datos de su representación visual, como también un mejor manejo de errores, y un mejor atributo de escalamiento en el sistema.

Por otra parte, se hace uso del framework Spring Boot para la creación del aplicativo web que funciona bajo la arquitectura cliente servidor, debido a la facilidad de control e implementación que tiene a la hora de implementar este tipo de arquitecturas.

Por ultimo se hace uso del protocolo STOMP (Text Oriented Messaging Protocol) que proporciona un formato de conexión interoperable para que los usuarios del protocolo puedan comunicarse bajo una arquitectura “Publish and subscribe” obteniendo un rendimiento optimo para la cantidad de procesamiento y peticiones que tendrá el proyecto.

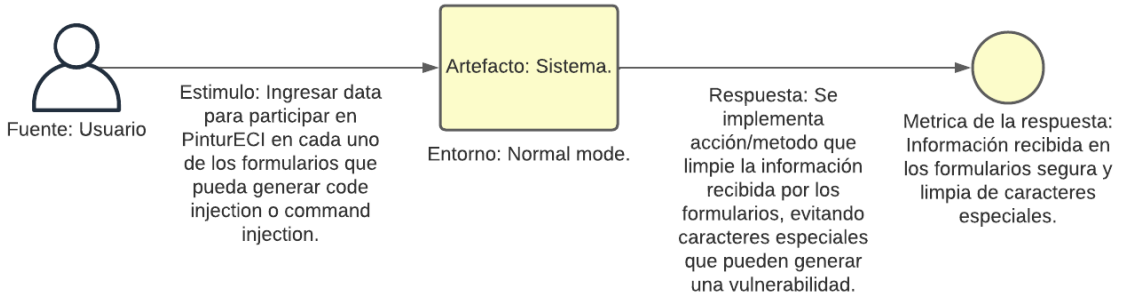
Atributo de calidad:

Disponibilidad:



- Estimulo: La caída de un servidor que contiene el servicio.
- Artefacto: Sistema.
- Entorno: Normal mode.
- Respuesta: El servicio continua normalmente debido a que la arquitectura está diseñada para escalar horizontalmente a través de un balanceador de cargas.
- Métrica de la respuesta: El servicio continúa funcionando a pesar de la caída de uno de los servidores, debido al balanceo de cargas que permite una amplia disponibilidad.

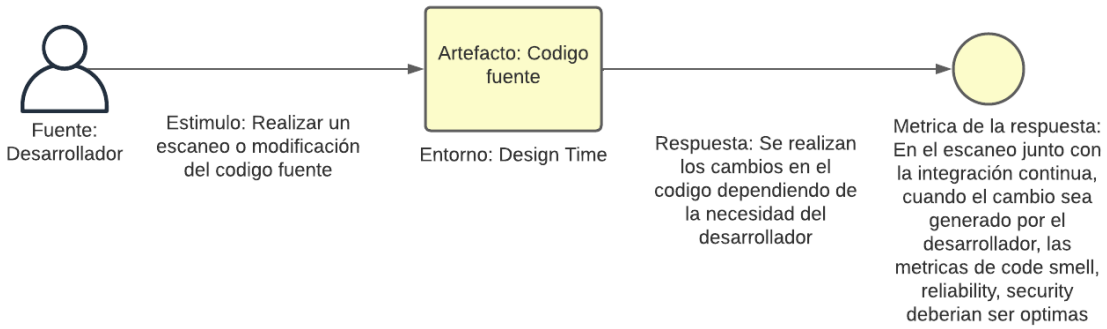
Seguridad:



- Estimulo: Ingresar data para participar en PinturECI en cada uno de los formularios que pueda generar code injection o command injection.
- Artefacto: Sistema.

- Entorno: Normal mode.
- Respuesta: Se implementa acción/método que limpie la información recibida por los formularios, evitando caracteres especiales que pueden generar una vulnerabilidad.
- Métrica de la respuesta: Información recibida en los formularios segura y limpia de caracteres especiales.

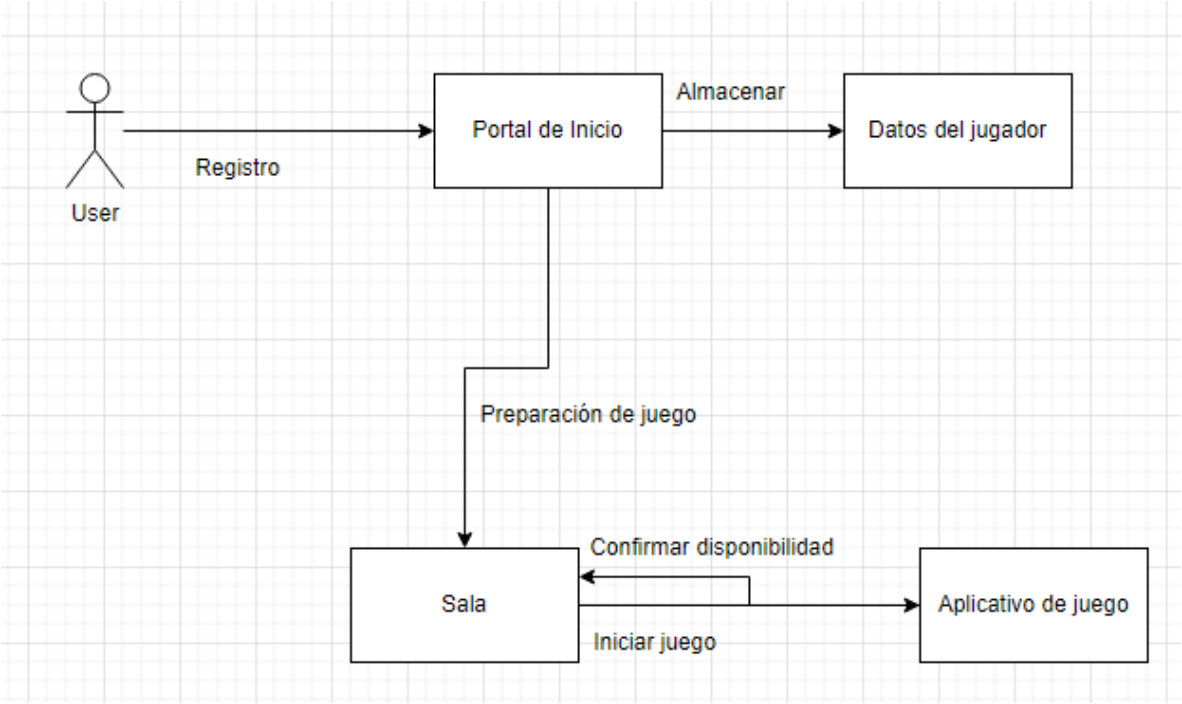
Mantenibilidad:



- Estimulo: Realizar un escaneo o modificación del código fuente.
- Artefacto: Código fuente.
- Entorno: Design Time.
- Respuesta: Se realizan los cambios en el código dependiendo de la necesidad del desarrollador.
- Métrica de la respuesta: En el escaneo junto con la integración continua, cuando el cambio sea generado por el desarrollador, las métricas de code smell, reliability, security deberían ser optimas.

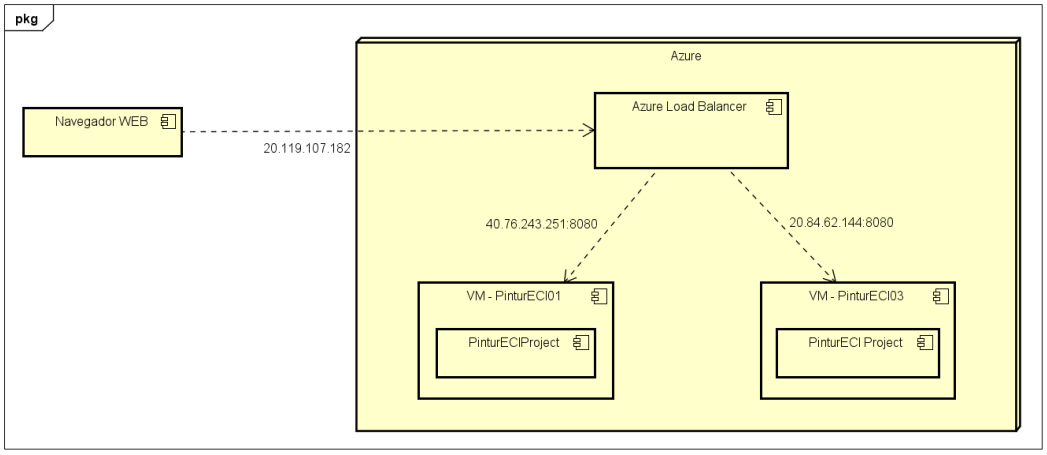
4. Vistas Arquitectónicas

Diagrama de contexto:



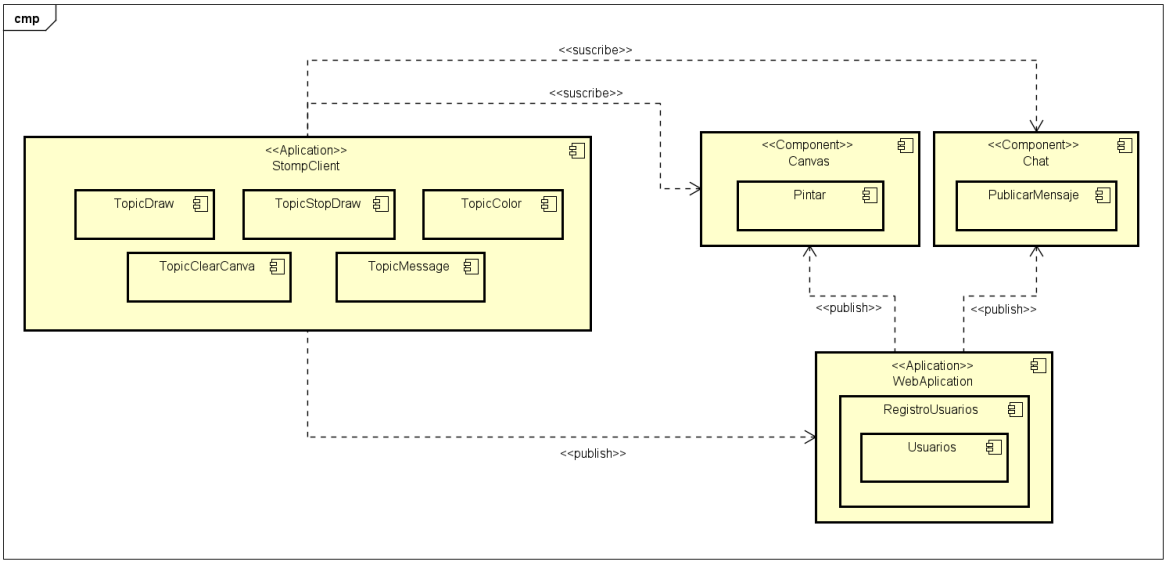
Dentro del diagrama de contexto tenemos nuestro producto final que es el aplicativo de juego donde cada usuario tiene un portal de registro sencillo y podrá acceder al servicio donde entrará a interactuar con los otros jugadores en PinturECI

Diagrama de despliegue:



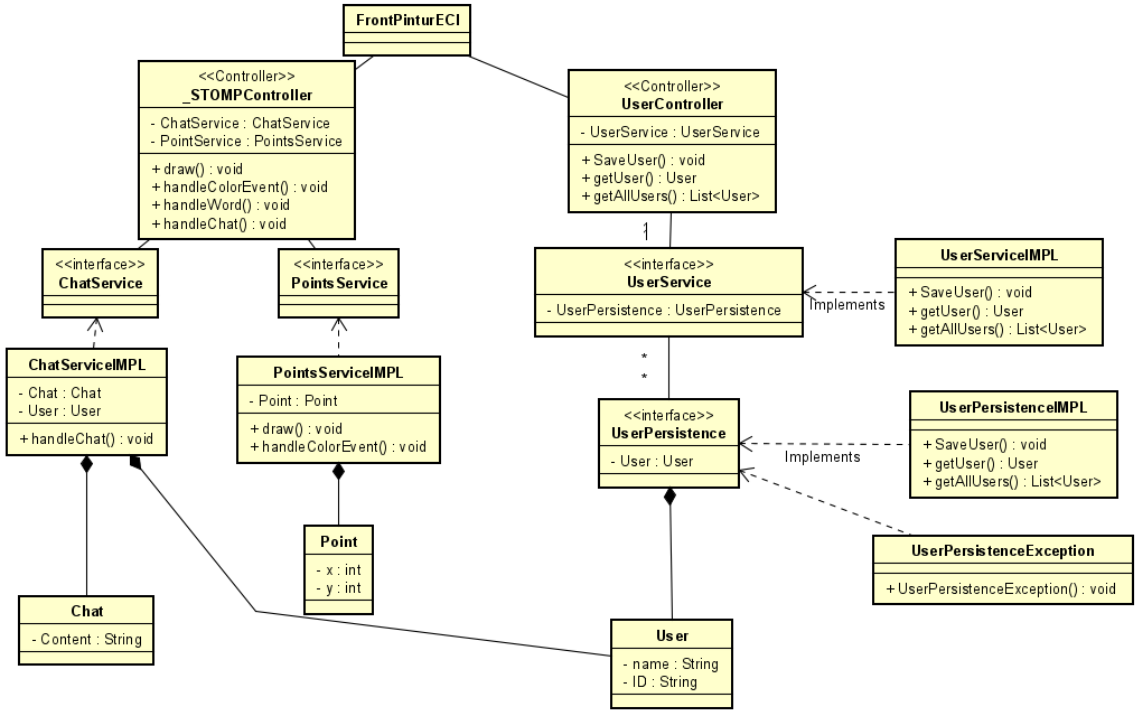
Visualiza el esquema de los elementos físicos que componen la solución. En pocas palabras hardware y configuraciones. En este caso se hace uso de dos maquinas virtuales que contienen el proyecto, y un balanceador de cargas que reparte las peticiones recibidas, obteniendo una mejor disponibilidad del sistema.

Diagrama de componentes:



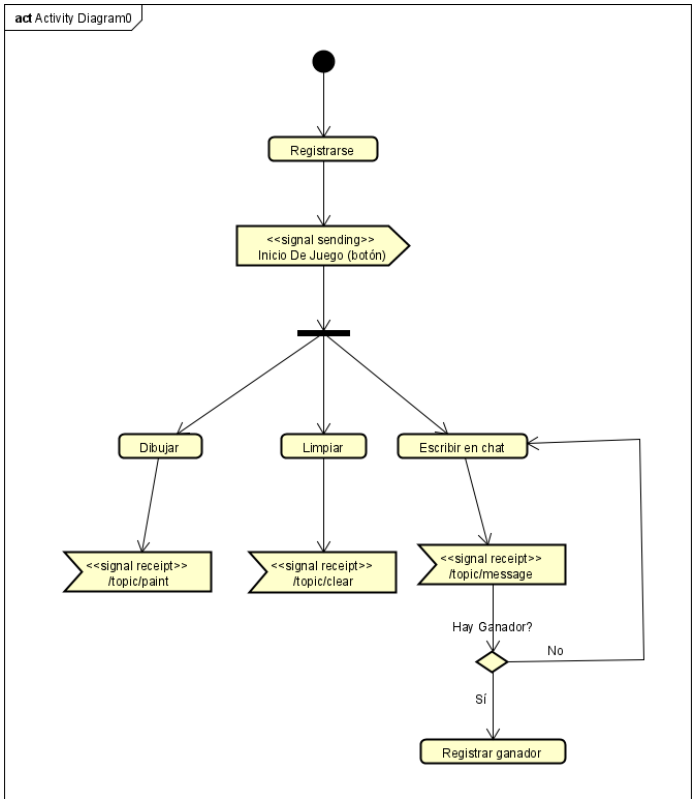
Representan las relaciones entre los componentes individuales del sistema mediante una vista de diseño estática, en este caso se ilustra la forma de comunicación de los componentes bajo los tópicos de chat, canvas, y el registro de usuarios.

Diagrama de clases:



Muestra la estructura de clases del sistema, sus propiedades, funcionalidades e interrelaciones entre ellas, en este caso cuenta con un userController que sirve para saber que jugadores entran en la sala y su ID para que puedan participar en el juego, cuenta con otro controller basado en STOMP que va a servir para el manejo en el servidor de todo lo relacionado con real time como la forma en que se pintan los puntos en el canva y el chat. Tiene una estructura MVC.

Diagrama de actividades:



Describe la descomposición del sistema en macroprocesos y como se comunican o interactúan. Este diagrama nos presenta el flujo de usuario en la aplicación y cuya finalización se logra al registrar que hubo algún ganador dentro del juego.

Conclusiones:

En el presente documento se abordaron los ítems a considerar en la estructuración de la arquitectura del aplicativo de PinturECI, la modelación de su arquitectura como también su manera de operación, explicada mediante diagramas y conceptos. El aplicativo funciona bajo estándares de los 3 atributos de calidad principales que en este caso son Disponibilidad mediante Azure funcionando con un balanceador de cargas, Mantenibilidad: con el software de SonarCloud y CircleCI, y Seguridad corrigiendo la vulnerabilidad de inyección de código y comandos.