Icesi-Health: estrategia de versionamiento, diseño de infraestructura y estrategia de Compilación

Valentina Arias Parra
José Luis Restrepo Obando
Jhon Alejandro Saldarriaga López

Universidad Icesi
Facultad de Ingeniería
Ingeniería Telemática
Cali
2023

Icesi-Health: estrategia de versionamiento, diseño de infraestructura y estrategia de Compilación

Valentina Arias Parra
José Luis Restrepo Obando
Jhon Alejandro Saldarriaga López

Automatización de la Infraestructura para Ingeniería Continua

Ing. Joan Sebastian Garcia Delgado
Ing. Juan Camilo Diaz

Universidad Icesi
Facultad de Ingeniería
Ingeniería Telemática
Cali
2023

Multinacional MyHealth

Es una empresa del sector salud con presencia en el país durante los últimos 30 años. MyHealth a lo largo de este tiempo, ha logrado consolidar y fidelizar una gran base de pacientes, alcanzando 150.000 aproximadamente. Este crecimiento, obligó a la compañía a evolucionar las soluciones de TI que soportan los procesos críticos de negocio. La empresa viene almacenando los registros de sus pacientes en un DB SQL, que no se ha podido actualizar, conectada a un BackEnd (Java), que ha venido operando de forma estable con una arquitectura monolítica. También sus reglas de negocio están escritas en Java.

Es por esto que se ha desarrollado una aplicación, denominada Icesi-Health que es una interfaz para un sistema de registros de pacientes. La aplicación se encuentra constituída por: un frontend en Node.JS Express, donde encontramos la interfaz de usuario que está programada utilizando librerías de código open source de JavaScript, CSS y HTML5 Canvas. Un backend donde encontramos la API servida por Node.js. Por último, una base de datos NoSQL denominada CouchDB. Por ahora, Icesi-Health ha evolucionado sus viejas reglas de negocio escritas en Java, y bajo una arquitectura monolítica, a una arquitectura microservicio desplegada en Node.Js.

La compañía también ha estado revisando el tema de la computación en la nube, vislumbrando la posibilidad de efectuar una posible migración de Icesi-Health. En la actualidad, todo su código se ejecuta en servidores on-premise. Sin embargo, algunos de sus arquitectos de software consideran importante, evolutivo y complementario acelerar el desarrollo de nuevas funciones de la aplicación y explorar las posibilidades que podría ofrecer el machine learning para aprovechar la base de información con la cuenta y que han acumulado por años.

Entrega #1: Versionamiento, infraestructura y compilación.

Proceso:

Para iniciar el análisis, se estudió el caso de trabajo, lo cual nos permitió entender la situación de la empresa, además de cómo estaba configurada la arquitectura inicial de la aplicación. Después de esto, empezamos a analizar qué tareas estaban detrás de los 2 requerimientos que nos pide el cliente: Automatizar el despliegue de la nueva versión de la aplicación Icesi-Health hasta un entorno productivo en la nube; y documentar las actividades.

Como equipo de trabajo, le garantizamos a nuestros clientes, los siguientes 3 items:

1. Yo como desarrollador quiero que el repositorio donde trabajo tenga una estrategia de versionamiento para asegurar la calidad y estabilidad de mi código.
    a. Explorar los tipos de estrategias de versionamiento que existen y que ventajas y/o desventajas tienen.
    b. Definir la estrategia de versionamiento que más se acomode al contexto.
    c. Definir las políticas de protección a la rama principal se van a usar
    d. Generar un Script que genere un repositorio dado el nombre y una plantilla junto con la estrategia de versionamiento implementado.
2. Nosotros como equipo de trabajo queremos que nuestra aplicación esté alojada en la nube para poder aprovechar los beneficios de escalabilidad y los servicios de machine learning que puede ofrecer un proveedor.
    a. Seleccionar un proveedor de nube que soporte/brinde servicios relacionados con ML.
    b. Diseñar la infraestructura necesaria para alojar nuestra aplicación (¿Qué servicios vamos a usar?, ¿Qué tipo de instancias son las adecuadas para implementar?, ¿Cuál nos ofrece mejor costo?)
3. Nosotros como equipo de trabajo queremos que nuestro código tenga una estrategia de compilación que nos permita entregar valor de forma rápida, confiable y de calidad.
    a. Explorar qué estrategias de compilación existen.
    b. Distinguir los diferentes ambientes de trabajo que vamos a usar.
    c. Configurar las dependencias que se van a usar en diferentes package.env.json
    d. Configurar los Scripts que van a correr cada uno de los ambientes
    e. Configurar el Dockerfile que va a desplegar el servicio.

Herramientas usadas:

Versionamiento: Git, Github y la estrategia Gitflow. Esta estrategia nos permite gestionar el flujo de un proyecto mediante un conjunto claro de ramas y reglas establecidas para manejarlas. Nos facilita la colaboración, el release de nuevas versiones, el control de errores y la implementación de ingeniería continúa al contar con ramas especializadas para integración del código y pruebas.

Para esta estrategia, se crean diferentes ramas que van a cumplir un papel en especial:

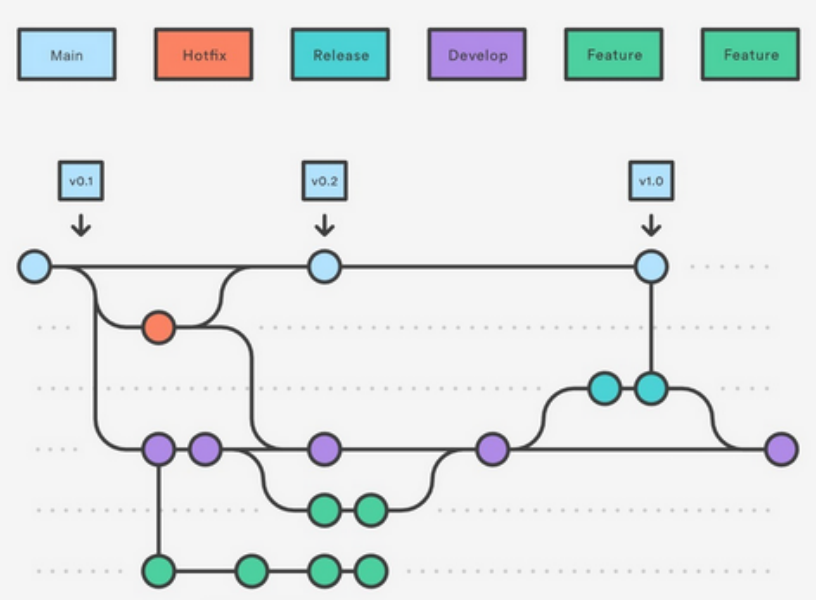

Ilustración 1. Estrategia de versionamiento Git Flow

Por otro lado, para las políticas de protección de rama, se escogió que la configuración más acorde sería:

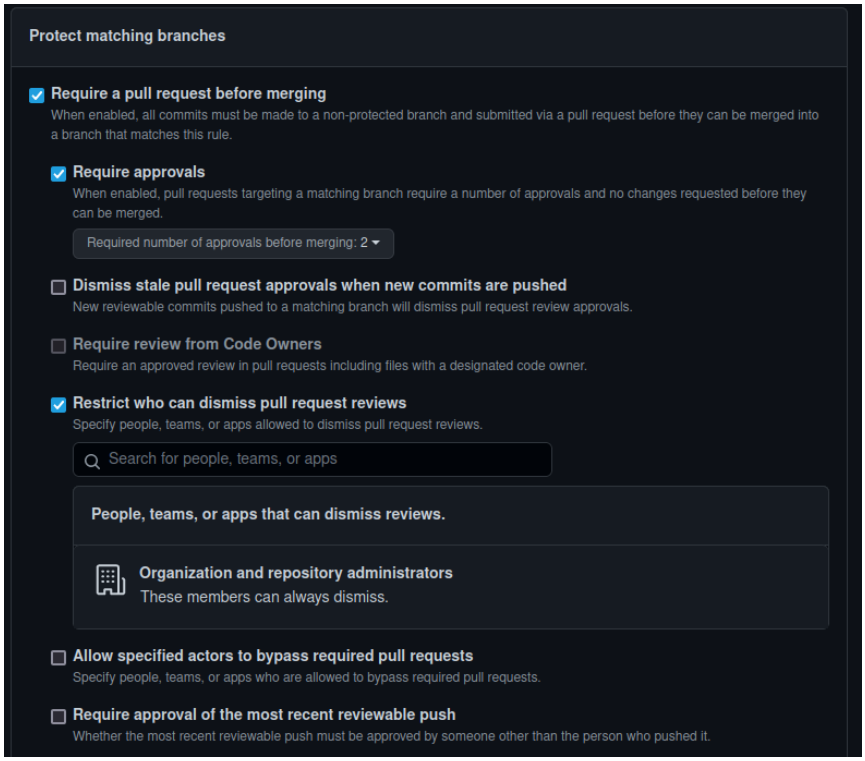

Ilustración 2. Políticas de protección de rama

Con esta configuración nos encargamos que solo se van a combinar cambios con otras ramas cuando estos cambios son revisados por al menos 2 personas.

- **Infraestructura**: Azure y Apps.diagram. Se eligió Azure debido a que este servicio de nube nos permite usar una cuota de prueba, al igual que ofrece muchos servicios acordes a la problemática, tal como balanceadores de carga, contenedores y Machine Learn.
- **Compilación**: Para definir una estrategia de compilación tuvimos en cuenta que existen diferentes herramientas y frameworks como:
    - TypeScript
    - Babel
    - Webpack
    - Gulp

Para esto, se eligió Webpack y se realizó la diferenciación de ambientes ya que cada ambiente hace uso de dependencias diferentes. Por lo tanto, se distingue los siguientes ambientes:

- **Developer**: instala las dependencias necesarias para continuar con el desarrollo del proyecto.
- **Testing**: Instala las librerías para realizar las pruebas unitarias del código (Jest o Mocha).

Finalmente, para desplegar la aplicación se decide usar contenedores, por lo cual se configura el dockerfile con los comandos necesarios para desplegar la imagen necesaria.

Retrospectiva del proyecto

¿Que salió bien?	¿Qué salió mal?	¿Cómo mejorar?
La estrategia de versionamiento fue relativamente sencilla de entender y configurar, debido a que solo requiere conocimiento de las herramientas de Git y Github, las cuales a lo largo de nuestra carrera las hemos usado mucho.	Pensar en el diagrama de la infraestructura fue frustrante y desgastador porque no tenemos conocimientos sólidos sobre Cloud. Por lo tanto, a pesar de que se buscó información por varias fuentes, fue difícil determinar si lo que se estaba diseñando estaba bien o no.	Si bien se sabe que el propósito del curso no es enfocarse en la nube, hubiera sido útil una ayuda adicional en cuanto a la creación de infraestructura, dando un repaso general de ciertos elementos más usados como contenedores, máquinas virtuales u otros elementos.

Tabla 1. Retrospectiva del proyecto