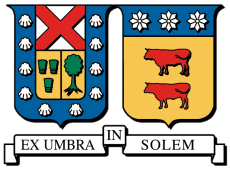
UTFSM

Departamento de Informática



Proyecto Entrega 2

Asignatura: INF331

Pruebas de Software

Profesora: Oscar Reyes H

Pablo Contreras, 201973572-1

Vania Gallardo, 201973619-1

Nobiembre-2024

Repositorio: www.github.com/Equipo-7-PDS-Proyecto/pruebas-software-frontend

1 Alcances de la herramienta

La herramienta Jenkins ha sido implementada principalmente en el backend del proyecto, con el objetivo de automatizar el proceso de integración continua (CI) y despliegue (CD). En esta etapa, Jenkins se ha configurado para realizar pruebas automatizadas sobre el código de la API, lo cual es fundamental para asegurar que las funcionalidades del backend funcionen correctamente antes de ser desplegadas en el entorno de producción.

A futuro, se planea extender el uso de Jenkins al frontend, lo cual se espera sea una tarea más sencilla dado que el frontend no cuenta con una capa de testeo.

2 Descripción del trabajo realizado

2.1 Frontend

En el front end se creo una nueva pagina del Perfil de Usuario donde están los datos Nombre, Email, Dirección y Teléfono. Estos datos vienen dados según el usuario que hace el login, los datos son editables mediante una consulta de PUT a la base de datos llamaba pachUser, que al hacer clic en editar se pueden editar los datos y al hacer clic en guardar, estos son guardados en la bd y mostrados en la página. Además, otro de los requerimientos fue el de mostrar en la navbar la página de Administrador solo si se ingresa como admin, si se ingresa como usuario esta opción no se encuentra disponible.

2.2 Backend

En el backend de la aplicación, se llevaron a cabo las siguientes modificaciones y mejoras funcionales:

- Nuevos campos en el modelo de usuario: Se añadieron los campos de "dirección" y "teléfono" al modelo de usuario. Esta información permite tener un perfil de usuario más completo.

- Endpoint para editar usuarios: Se desarrolló un nuevo endpoint que permite editar los datos de un usuario existente, facilitando la actualización de información clave, como dirección y teléfono.

- Ampliación de información en el login: Se modificó el endpoint de login para incluir en la respuesta la información adicional del usuario (dirección y teléfono), proporcionando datos más completos al momento de la autenticación.

2.3 Integración de Jenkins, Testeo y Hooks

Además de las mejoras en la funcionalidad del backend, se incorporó Jenkins como herramienta de integración continua (CI) y despliegue continuo (CD), lo que permite automatizar el proceso de compilación, testing y despliegue de la aplicación. Esta integración incluye los siguientes aspectos:

- Automatización del Testeo: Jenkins ejecuta una batería de pruebas automatizadas en cada cambio del código. Para esto, se configuró Mocha como framework de testing, con reportes en formato JUnit para facilitar el monitoreo y análisis de los resultados.

- Hooks con Slack y GitHub: Se configuraron hooks que notifican en Slack cada vez que se ejecuta un proceso en Jenkins (éxito o fallo). Además, se integró un webhook en GitHub para que cada push o merge en la rama principal dispare automáticamente el pipeline en Jenkins, asegurando que cada cambio pase por el proceso de CI/CD antes de ser desplegado.

2.4 Especificar dependencias entre la herramienta y la aplicación

Node.js y npm: Jenkins depende de Node.js y npm para instalar y ejecutar las dependencias de JavaScript del proyecto. Durante la etapa de construcción, Jenkins utiliza npm install para asegurar que todas las dependencias del proyecto estén disponibles antes de compilar y ejecutar las pruebas.

Mocha: Jenkins utiliza Mocha como framework de testing para ejecutar las pruebas automatizadas del backend.

PM2 (o Systemd para despliegue): Para el despliegue en producción, se configuró systemd para manejar la ejecución de la aplicación, garantizando que el backend se mantenga activo y reinicie automáticamente en

caso de fallas. Aunque inicialmente se consideró el uso de PM2, se optó por systemd para mayor estabilidad en el entorno de Jenkins.

Slack (Notificaciones): Jenkins se configura para enviar notificaciones a Slack en cada etapa del pipeline (éxito o fallo), lo cual facilita la supervisión del estado de las compilaciones y despliegues. Esta integración asegura que el equipo esté informado en tiempo real sobre cualquier cambio o problema en el proceso de CI/CD.

3 Uso de Jenkins

Jenkins se implementó como herramienta de Integración Continua y Despliegue Continuo (CI/CD) en el proyecto, permitiendo automatizar y gestionar los procesos de compilación, testing y despliegue del backend de la aplicación. Esta sección detalla el flujo de trabajo y configuración realizada en Jenkins para cumplir con estos objetivos.

3.1 Configuración del Pipeline

Se configuró un pipeline en Jenkins para el backend, estructurado en varias etapas que se ejecutan de manera secuencial:

- Instalación de Dependencias: En esta etapa, Jenkins instala todas las dependencias necesarias para el proyecto utilizando npm install. Esto asegura que siempre se usen las versiones de librerías requeridas.

- Compilación de la Aplicación: Una vez instaladas las dependencias, se compila el código TypeScript a JavaScript usando npm run build. Esta compilación convierte el código fuente a un formato ejecutable para Node.js en producción.

- Ejecución de Pruebas Automatizadas: Jenkins ejecuta una batería de pruebas automatizadas usando Mocha. Los resultados se generan en formato JUnit, lo que permite a Jenkins visualizarlos y determinar si el código pasa todas las pruebas establecidas. Esta etapa es fundamental para verificar que los cambios en el código no rompan la funcionalidad existente.

- Despliegue en el Servidor: Si las pruebas son exitosas, Jenkins despliega el backend en el servidor de producción mediante systemd, lo que permite manejar el proceso de la aplicación de forma robusta y reiniciarla automáticamente en caso de fallo. Para esto, se configuró un servicio de systemd que Jenkins puede detener y reiniciar durante el despliegue.

3.2 Notificaciones y Webhooks

Para mejorar la comunicación y la colaboración, se configuraron notificaciones automáticas:

- Slack: Jenkins envía notificaciones a un canal de Slack en cada ejecución del pipeline, indicando si el proceso fue exitoso o falló. Esto permite al equipo estar al tanto del estado de la integración y reaccionar rápidamente ante cualquier problema.

- GitHub: Se configuró un webhook en GitHub que dispara automáticamente el pipeline en Jenkins cada vez que hay un push o merge en la rama principal. Esto asegura que cualquier cambio en el código pase por el proceso de CI/CD, reduciendo los riesgos de introducir errores en producción.

4 Problemas encontrados y soluciones

En la entrega pasadas las consultas funcionaban en la base de datos pero no al aplicarlas en el front end. Ahora si funciona la conexión mediante la Api y las consultas que se hacen en el front son recibidas por el back.

Inicialmente, se utilizó PM2 como gestor de procesos para el despliegue del backend en el servidor. PM2 es una herramienta popular para ejecutar aplicaciones de Node.js en producción, pero se presentaron problemas significativos en el contexto de Jenkins. PM2 no estaba bien documentado para integrarse con Jenkins y presentó dificultades debido a cómo Jenkins maneja los subprocesos. Esto generó problemas de permisos y de persistencia del proceso, haciendo que el despliegue fallara o que el backend no se mantuviera en ejecución después de completarse el pipeline.

Para solucionar estos problemas, se decidió cambiar el gestor de procesos a systemd. systemd permitió gestionar el backend como un servicio del sistema, lo que asegura que se mantenga en ejecución incluso si el proceso de Jenkins finaliza. Esta solución mejoró significativamente la estabilidad del despliegue y simplificó la configuración, eliminando la necesidad de configuraciones adicionales que PM2 requería en este contexto.