UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Documentación
Web de tienda en línea
Fashion Trend

Integrantes	Denyss Carcamo	202073035-0
	Gastón Quevedo	201830002-0
	Jorge Roldán	202073116-0
Fecha	17/06/2024	
Profesor	Oscar Reyes H.	
Asignatura	Pruebas de Software	
Grupo	5	

INF-331

Índice

1. Introducción

La tienda en línea "Fashion Trend"se especializa en la venta de ropa y accesorios de moda, pero ha enfrentado varios desafíos que afectan tanto la experiencia del cliente como la eficiencia operativa interna. Los clientes frecuentemente tienen dificultades para encontrar productos específicos y obtener información detallada, lo que impacta negativamente en la satisfacción del cliente y las ventas. Además, la gestión manual del inventario y los detalles de los productos ha resultado ser un proceso lento y propenso a errores.

Para abordar estos problemas, "Fashion Trend"ha decidido desarrollar un sistema de gestión de inventario y catálogo en línea. Este sistema permitirá a los clientes buscar productos por categoría, tipo, tamaño y color, además de ver la disponibilidad de productos en tiempo real y obtener detalles completos sobre cada artículo, incluyendo imágenes, descripciones y precios.

El sistema también ofrecerá una interfaz de administración para el equipo de la tienda. Desde esta interfaz, los empleados podrán agregar nuevos productos al catálogo, editar detalles existentes, gestionar cantidades en inventario y eliminar elementos que ya no estén disponibles. Esta funcionalidad mejorará significativamente la eficiencia operativa y reducirá los errores asociados con la gestión manual.

2. Tecnologías

En el panorama digital actual, es crucial desarrollar una aplicación web escalable y de alto rendimiento para manejar las crecientes demandas de los usuarios y garantizar una experiencia de usuario perfecta. La elección de tecnologías como MongoDB, Fastify, React y Jest garantiza que la aplicación no solo sea eficaz, sino también mantenible y fácil de ampliar a medida que evolucionan los requisitos.

- **MongoDB:** Es una base de datos NoSQL conocida por su escalabilidad y flexibilidad. Utiliza un modelo sin esquema, que permite un rápido desarrollo e iteración de aplicaciones. MongoDB almacena datos en un formato similar a JSON (BSON), lo que lo convierte en una opción natural para las aplicaciones web modernas y permite una integración perfecta con backends y frontends basados en JavaScript. El alto rendimiento de la base de datos se logra mediante funciones de indexación, replicación y equilibrio de carga, lo que la hace adecuada para aplicaciones que necesitan manejar grandes volúmenes de datos y altas tasas de transacciones.

- **Fastify:** Es un framework moderno para Node.js que enfatiza la velocidad y los bajos gastos generales. Fastify, conocido por su alto rendimiento, está diseñado para manejar una gran cantidad de solicitudes de manera eficiente, lo que lo hace ideal para el desarrollo backend donde el rendimiento es crítico. La arquitectura de plugins

de Fastify permite una amplia personalización y extensión de la funcionalidad sin sacrificar el rendimiento.

- **React:** Es una biblioteca de JavaScript popular para crear interfaces de usuario, particularmente aplicaciones de una sola página donde las actualizaciones de datos en tiempo real son cruciales. La arquitectura basada en componentes de React promueve la reutilización y la modularidad, lo que permite a los desarrolladores crear interfaces de usuario complejas componiendo componentes más pequeños y aislados. El uso de un DOM virtual mejora el rendimiento al actualizar solo las partes de la interfaz de usuario que han cambiado, en lugar de volver a representar toda la página. El sólido apoyo de la comunidad de React y su extenso ecosistema de bibliotecas y herramientas lo convierten en una opción confiable para el desarrollo frontend.

- **Jest:** Es un framework de prueba integral para JavaScript, diseñado para garantizar la calidad y confiabilidad del código. Funciona de inmediato con una configuración mínima, lo que permite a los desarrolladores escribir y ejecutar pruebas rápidamente. Jest es conocido por su velocidad y confiabilidad, con características como la ejecución de pruebas en paralelo y la observación inteligente de pruebas para optimizar el proceso de prueba. Admite pruebas de instantáneas, que capturan la salida renderizada de los componentes de la interfaz de usuario para realizar un seguimiento de los cambios a lo largo del tiempo. Las sólidas capacidades de mocking de Jest y su extensa biblioteca de aserciones lo convierten en una herramienta poderosa para pruebas unitarias y de integración.

- **Jenkins:** Servidor que ayuda en la automatización de parte del proceso de desarrollo de software utilizando integración continua. Facilita aspectos de la entrega continua admitiendo herramientas de control de versiones, secuencias de comandos de consola y programas por lotes de Windows.

3. Alcances de la herramienta

- Catálogo de productos:
  - Vista principal con todos los productos: Se muestra cada producto con su nombre, su precio y una foto que lo acompaña.
  - Vista particular con detalles de cada producto: Indica cada una de las características del producto.
- Gestión de inventario:
  - Añadir, editar y eliminar: Los empleados que hayan iniciado sesión pueden añadir productos, editar los ya existentes o eliminar alguno.

- Visualización y búsqueda de productos registrados: Los productos son mostrados en una lista y pueden ser encontrados con una barra de búsqueda.

- Gestión de Usuarios:
  - Creación usuarios con distintos roles: Se tienen 3 roles, cada uno con funciones diferentes.
  - Rol Administrador: Es capaz de crear empleados, ver una lista completa de los usuarios registrados, editarlos o eliminarnos.
  - Rol Empleado: Pueden agregar, editar o eliminar productos, también pueden ver una lista con cada uno de los productos existentes y buscarlos con la barra de búsqueda.
  - Rol Cliente: Pueden ver los productos del catálogo y no tienen permiso para modificarlos de ninguna forma.

- Carro de compra:
  - Visualizar, añadir y eliminar: Los clientes que hayan iniciado su sesión pueden visualizar sus productos, añadir productos a su carro de compra, o eliminar alguno ya agregado anteriormente.

4. Descripción

4.1. Backend

El desarrollo del backend consta de tres tareas principales: inicialización, desarrollo de funciones principales y refinamiento de funciones principales. Cada tarea se basa en la anterior para garantizar un sistema backend robusto y escalable.

1. **Inicializar el servidor**

   a) Configurar la estructura del proyecto: Crear la estructura de carpetas para el backend. Compuesta principalmente por modelos, controladores, rutas, y middleware.

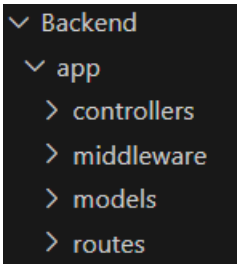


Figura 1: Estrcutura del backend

b) Instalar Fastify y otras dependencias: Instalar Fastify y otras dependencias necesarias para el proyecto.

Figura 2: Dependencias del backend

c) Crear archivo de servidor: Crear el archivo del servidor principal y configurarlo para iniciar el servidor.

d) Configurar variables de entorno: Configurar variables de entorno para la configuración del servidor.

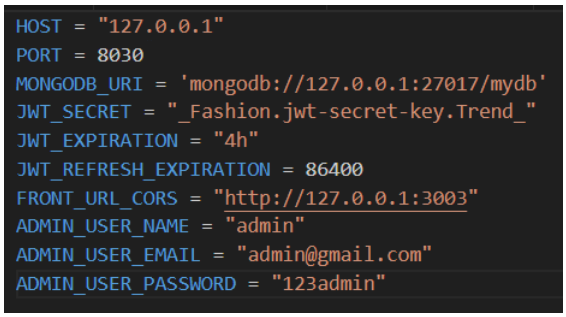


Figura 3: Ejemplo de variables de entorno

e) Conectarse a MongoDB: Configurar una conexión a MongoDB usando Mongoose.

f) Inicializar usuario administrador y roles: Crear el usuario administrador inicial y definir los roles de los usuarios (user, employee, y admin).

2. **Desarrollar funciones principales**

a) Registrar plugins para Fastify: Registrar los complementos (plugins) necesarios para que Fastify amplíe su funcionalidad.

b) Crear modelos: rol, refresh token, usuario, producto, e imagen

c) Implementar autenticación: Implementar la autenticación de usuarios mediante JSON web tokens (JWT) y tokens de actualización.

d) Desarrollar APIs CRUD de usuario y producto: Crear APIs CRUD (Create, Read, Update, Delete) para administrar usuarios y productos.

e) Crear rutas de usuario, producto, y autenticación: Desarrollar las rutas para las distintas operaciones que hay sobre los usuarios, productos, y autenticación (registro, inicio de sesión y actualización de tokens).

f) Implementar hook para rutas de usuarios y productos: Utilizar middleware como hook para proteger las rutas de usuarios y productos al requerir encabezados de autorización y validar tokens JWT.

3. **Refinar funciones principales**

a) Implementar Role-Based Access Control (RBAC): Mejorar la seguridad de la aplicación implementando un control de acceso basado en roles para restringir el acceso a ciertas API según los roles de los usuarios.

b) Mejorar el manejo de errores: Mejorar el manejo de errores para proporcionar mensajes de error más informativos y garantizar respuestas de error consistentes.

4. **Mejorar aplicación**

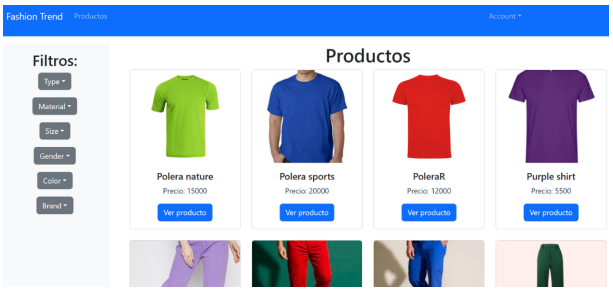
a) Desarrollar APIs relacionadas al carro de compra: Crear APIs relacionadas al carro de compra para agregar o eliminar productos en este, y obtenerlos.

b) Crear rutas: Desarrollar las rutas para las operaciones relacionadas al carro de compra.

c) Mejorar README: Crear archivo README para indicar instrucciones básicas.

4.2. Frontend

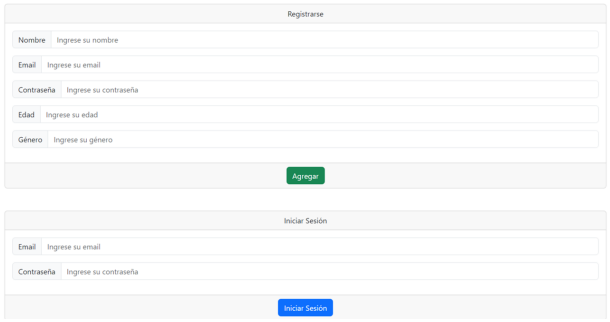
Tenemos una vista principal, a la izquierda están los filtros por cada característica de los productos y a la derecha se muestra cada producto con su nombre y su precio:



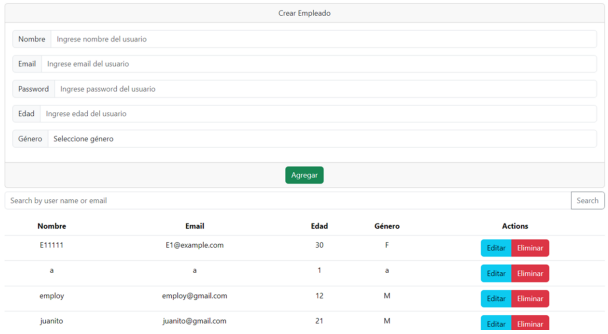
Al apretar el botón 'ver producto' se accede a una vista que contiene los detalles de este y su imagen en un mayor tamaño para que el cliente tenga la información necesaria para realizar una compra informada:

En la pantalla de detalle de producto existen botones para agregar el producto al carrito de compras y para borrarlo.

Desde la barra de navegación se puede acceder al login, en este se encuentra un formulario para registro y otro para inicio de sesión, este registro genera cuentas con el rol de cliente:

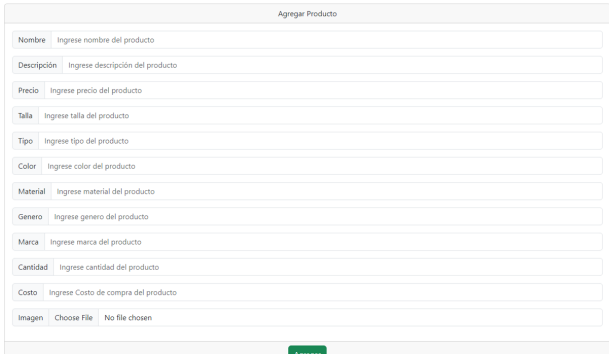


Si un administrador hace login es dirigido a una vista de administrador, en esta puede crear cuentas para empleados, ver todos los usuarios existentes y utilizar una barra de búsqueda para encontrar alguno:

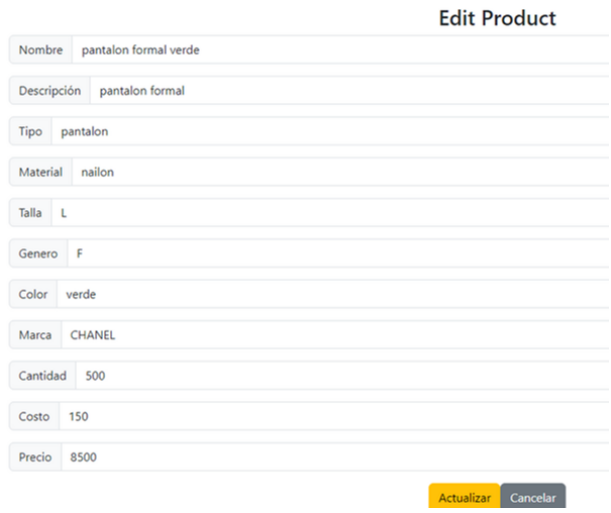


Por cada usuario tiene la opción de eliminarlo o editarlo en caso de ser necesario:

Si un empleado realiza login tiene la habilidad de registrar nuevos productos en el sistema indicando cada una de sus características y una imagen:



Estos productos se pueden visualizar en una lista y cada uno tiene la opción de ser eliminado o actualizar alguno de sus campos mediante un formulario que es desplegado al apretar el botón 'actualizar'. En esta página también se encuentra una barra de búsqueda que permite encontrar productos más fácilmente:



4.3. Dependencias herramienta con aplicación

La infraestructura de nuestra aplicación se dividió en tres partes, base de datos, backend y frontend.

1. **Base de Datos:**
   - *a*) Herramienta: MongoBD
   - *b*) Descripción: Almacenamiento de la aplicación, aquí se guarda toda la información necesaria para el funcionamiento de la aplicación
   - *c*) Pruebas: Aunque Jest ofrece la opción de hacer pruebas directas a MongoBD se decidió realizar pruebas a un siguiente nivel, ya que se considero que las pruebas del Backend cubrían lo necesario para probar también la base de datos.
2. **Backend:**
   - *a*) Herramienta: Fastify

b) Descripción: Sección encargada del manejo de la información, tanto almacenada como a almacenar, es la logica de la aplicación.

c) Pruebas: Se hicieron suites de pruebas en este nivel, considerando las conexiones al nivel anterior.

3. **Frontend:**

a) Herramienta: React

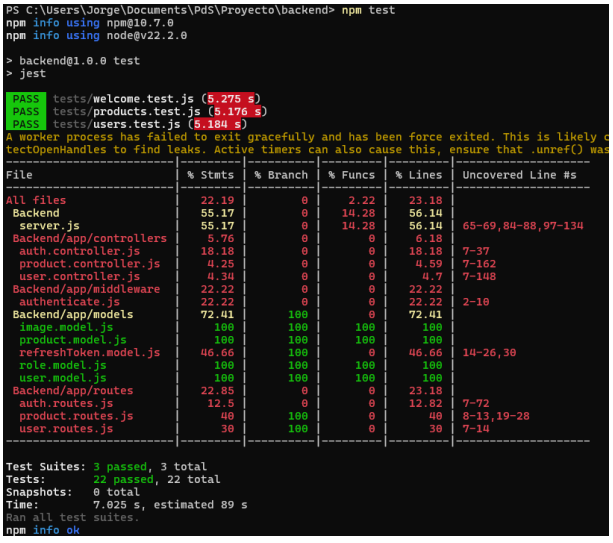
b) Descripción: Encargada de entregar una interfaz para permitir la interacción con el usuario

c) Pruebas: Actualmente no hay pruebas implementadas en esta sección, pero se encuentran planificadas y en desarrollo.

5. Testing

El desarrollo del testing se dividió en dos partes, backend y frontend. En la sección de backend se uso Jest combinado con Fastify para realizar las pruebas, en el transcurso se probaron diversos métodos, funciones y librerías auxiliares que permitieran ejecutar las pruebas, en lo cual se uso finalmente el método de inyección otorgado por Fastify, el cual consiste entregar el método y el cuerpo de la solicitud HTTP y la librería comprueba su ejecución, pero estas ejecuciones no tienen ninguna repercusión fuera de la suite, solo son validas dentro.

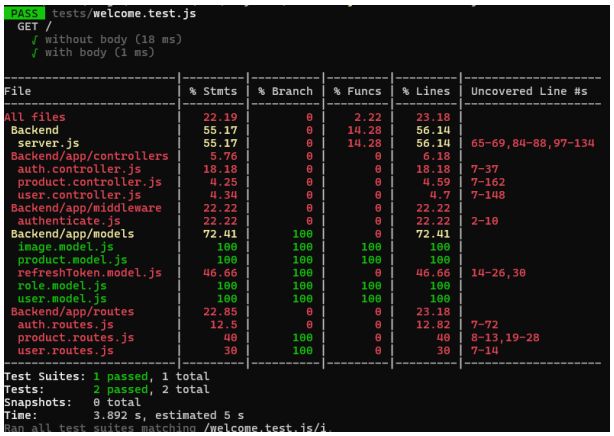
En este proceso se desarrollaron tres suite de casos de pruebas, la primera enfocada en el endpoint inicial, la segunda en el endpoint de productos y finalmente el endpoint de usuarios.



En la figura se aprecia la ejecución de las tres suites, además se muestra una tabla que corresponde a la ejecución de las pruebas en todos los archivos, mostrando que lineas

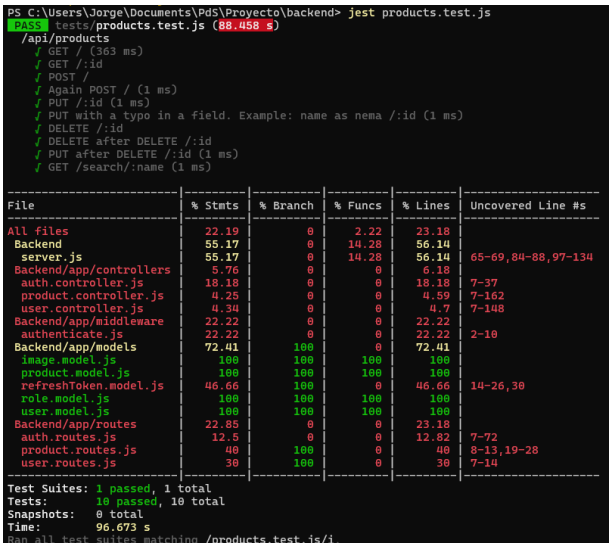
no son ejecutadas , porcentaje de ejecución, funciones, entre otra información que puede resultar útil al momento de analizar las pruebas

Para el endpoint inicial existían dos casos posibles, que la petición viniera con un cuerpo o no, y justamente eso fueron los casos desarrollados en esta suite.

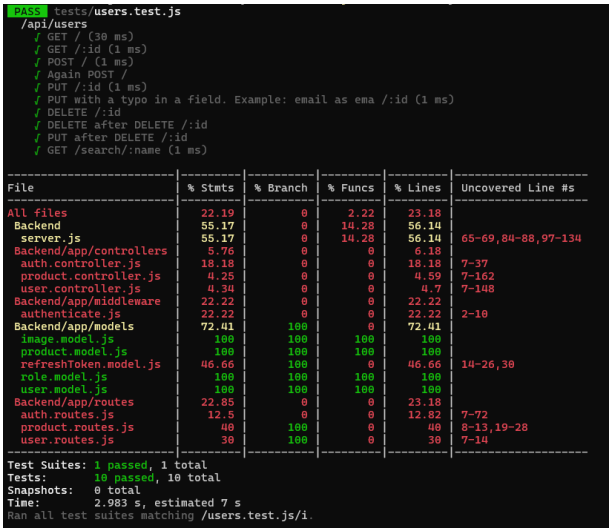


Para el endpoint de productos se realizaron mas pruebas, a continuación se describen brevemente cada caso ejecutado:

1. Se obtienen todos los productos.
2. Se obtiene un producto en base a su id.
3. Se sube un producto.
4. Se sube el mismo producto, se busca que sea rechazado por ya existir.
5. Se actualiza un producto.
6. Se actualiza un producto pero compitiendo un error en su formulario.
7. Se borra un producto.
8. Se borra un producto ya eliminado.
9. Se actualiza un producto ya eliminado.
10. Se busca un producto por su nombre



Para la ultima suite no existe mucha diferencia a nivel descriptivo con la anterior, la diferencia radica en que los datos a rellenar son distintos porque usuarios y productos son entidades distintas.



6. Jenkins

El uso de Jenkins partió con la instalación local en uno de nuestros computadores, junto con la instalación de ngrok para poder manejar los webhooks y permitir el acceso al resto de integrantes del grupo.

La primera integración realizada fue con github, donde se configuro un webhook dentro del repositorio y dentro de Jenkins se uso el plugin de git, con el cual cada vez que detectaba un cambio en la rama develop corría el pipeline descrito en el repositorio.

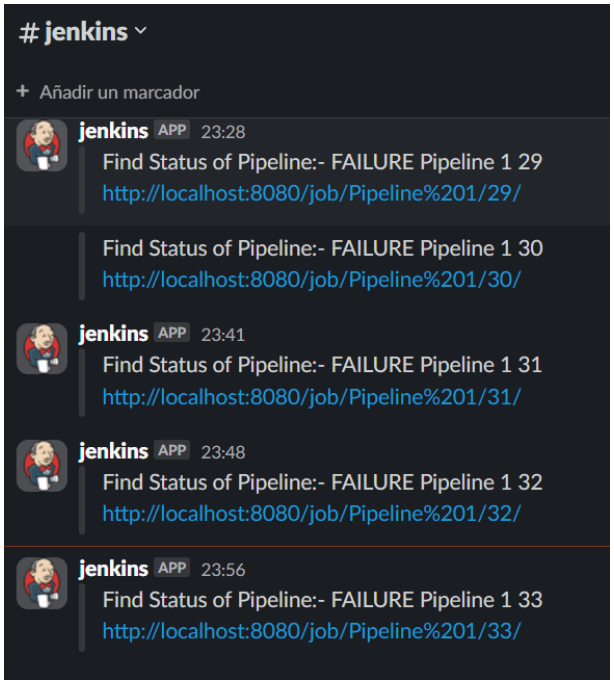
El pipeline descrito fue el siguiente:

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                dir('Backend') {
                    bat 'npm install'
                }
                dir('frontend/my-react-app') {
                    bat 'npm install'
                }
            }
        }
        stage('Test'){
            steps {
                dir('Backend') {
                    bat 'npm test'
                }
                dir('frontend/my-react-app') {
                    bat 'npm test'
                }
            }
        }
        stage('Deploy') {
            steps {
                dir('Backend') {
                    bat 'npm start'
                }
                dir('frontend/my-react-app') {
                    bat 'npm start'
                }
                input message: 'Finished using the web site? (Click "Proceed" to continue)'
                dir('Backend') {
                    bat 'taskkill /IM node.exe /F'
                }
                dir('frontend/my-react-app') {
                    bat 'taskkill /IM node.exe /F'
                }
            }
        }


    }
    post {
        always {
        //Add channel name
        slackSend channel: 'jenkins',
        message: "Find Status of Pipeline:- ${currentBuild.currentResult} ${env.JOB_NAME} ${env.BUILD_NUMBER} ${BUILD_URL}"
        }
    }
}
```

Donde contamos con 3 fases esenciales, primero se instalan todas las dependencias, segundo se ejecuta cada prueba y por ultimo se monta la aplicación. Adicionalmente a estas 3 fases con cada ejecución se llama a un post, que mediante el plugin de Jenkins y la integración de Slack envía un mensaje al canal destinado.



En si no fue un proceso simple, ocurrieron bastantes dificultades para configurarlo e

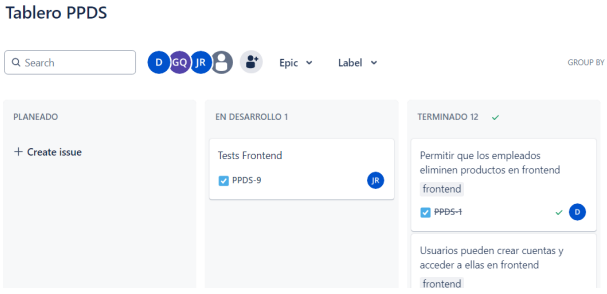
integrarlo, pero se logro que hiciera un testeo automatizado cuando la rama se actualizara permitiendo conocer que lo que ya llevábamos hecho siguiera funcionando.

A continuación se muestra una ejecución de Jenkins

Donde el fallo ocurrió que al momento de montar la aplicación la instancia local de Jenkins no contaba con .env actualizado de la app.

7. Problemas encontrados y resueltos

1. **Organización:** El primer problema que tuvimos fue al no organizar las tareas que realizaríamos desde un inicio, esto llevó a descoordinaciones y desconocimiento sobre cómo iba el progreso del desarrollo en cada uno de los miembros del equipo. La solución fue empezar a utilizar jira, escribiendo las tareas que serían necesarias y asignándolas a cada uno de los integrantes del equipo, estas fueron organizadas en las categorías: planeado, en desarrollo y terminado. Solucionando nuestro problema de organización.



2. **Interacción entre Herramientas:** Nuestro segundo problema fue una interacción que ocurria entre nuestras herramientas involucradas, Jest, Factify y la extracción de datos de un archivo de variables de entorno (.env). Esto debido a la naturaleza de pruebas pequeñas de Jest, el cual busca la ejecución rápida de pruebas a componentes e interacciones, para lo cual siempre sugiere el uso de Mocks para omitir solicitudes e interacciones y simplemente simularlas, esto ultimo fue la causa del problema, Factify y las variables de entorno causaban bastantes llamadas asincrónicas que Jest no estaba esperando, investigando sobre esto se cambio la configuración de Jest para que reconociera la extracción de variables de entorno y las esperara y se uso una función especial de Jest que es un mock de temporizadores, inserta un temporizador falso que obliga a Jest a respetar los tiempo y nos permitió correr nuestras suites de casos de prueba correctamente.

3. **Jenkins** Nuestro tercer problema fue con Jenkins, ya que por desconocimiento de la herramienta tuvimos varios percances a la hora de instalarla y configurarla, ademas de sus implementaciones. Pero con estudió del tema y a base de muchos ejercicios de prueba y error logramos automatizar el proceso de instalación, pruebas y montaje de la app

8. Administración

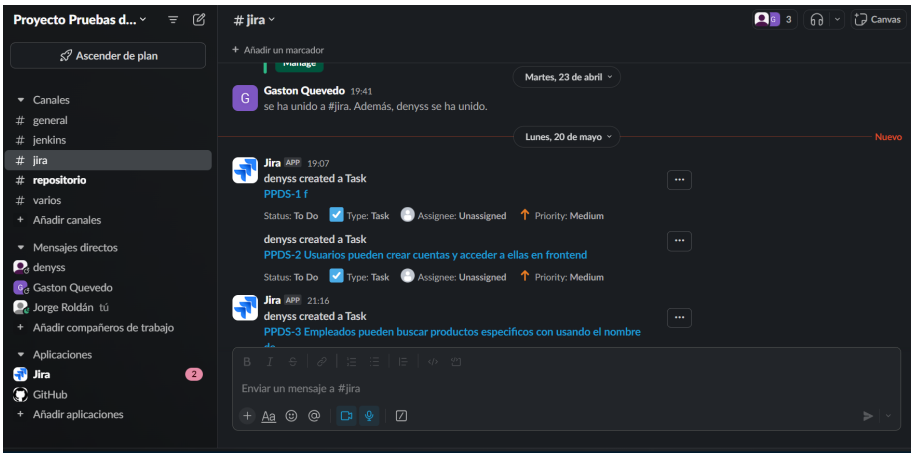
Se conto con un servidor de Slack para notificarnos de los avances del proyecto, este mismo contaba con integraciones al repositorio en github, al tablero de Jira y a las ejecuciones de Jenkins.

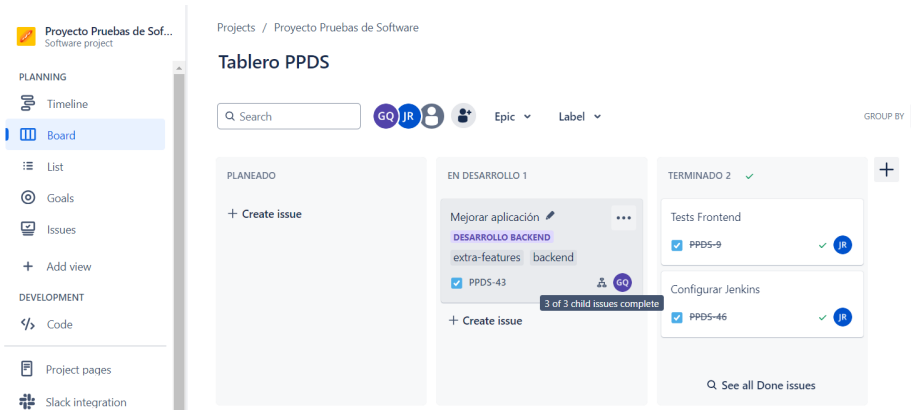
- **Slack:** Aplicación de mensajería diseñada para la comunicación y colaboración en equipo. Admite mensajería directa, chats grupales y canales organizados por temas, proyectos o equipos, lo que mejora la eficiencia de la comunicación. Slack permite compartir archivos sin problemas dentro de las conversaciones, fomentando entornos de trabajo colaborativos. Se integra con muchas otras herramientas, incluidas JIRA, GitHub y Jenkins, para enviar notificaciones y actualizaciones directamente dentro de los canales relevantes. La aplicación también cuenta con sólidas capacidades de búsqueda para localizar mensajes, archivos y enlaces rápidamente.

- **GitHub:** Plataforma ampliamente utilizada para control de versiones y codificación colaborativa, que utiliza Git. Permite a los equipos administrar repositorios de código, realizar un seguimiento de los cambios y trabajar juntos en proyectos de código. Las características clave incluyen solicitudes de extracción, revisiones de código y problemas, que facilitan la colaboración y mantienen la calidad del código.

- **Jira:** Herramienta de seguimiento de problemas y gestión de proyectos ampliamente utilizada por los equipos de desarrollo de software. Permite a los equipos crear, asignar, rastrear y gestionar tareas y proyectos de manera eficiente. Con un fuerte soporte para metodologías ágiles como Scrum y Kanban, JIRA facilita la planificación de sprints, la gestión de trabajos pendientes y proporciona herramientas como gráficos de evolución para monitorear el progreso.

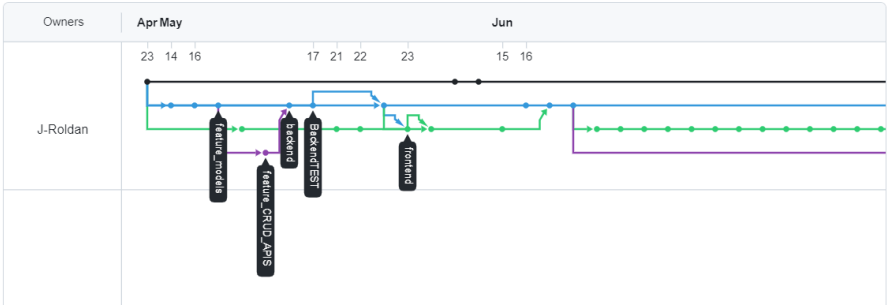
- **Jenkins:** Servidor de automatización de código abierto que se utiliza principalmente para crear, probar e implementar software. Permite la automatización de varias etapas del ciclo de vida del desarrollo de software a través de su extenso ecosistema de complementos, que admite integraciones con herramientas como GitHub, JIRA y Slack. Jenkins facilita la creación de canales de CI/CD complejos, lo que permite a los equipos automatizar sus procesos de construcción, prueba e implementación.





Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.