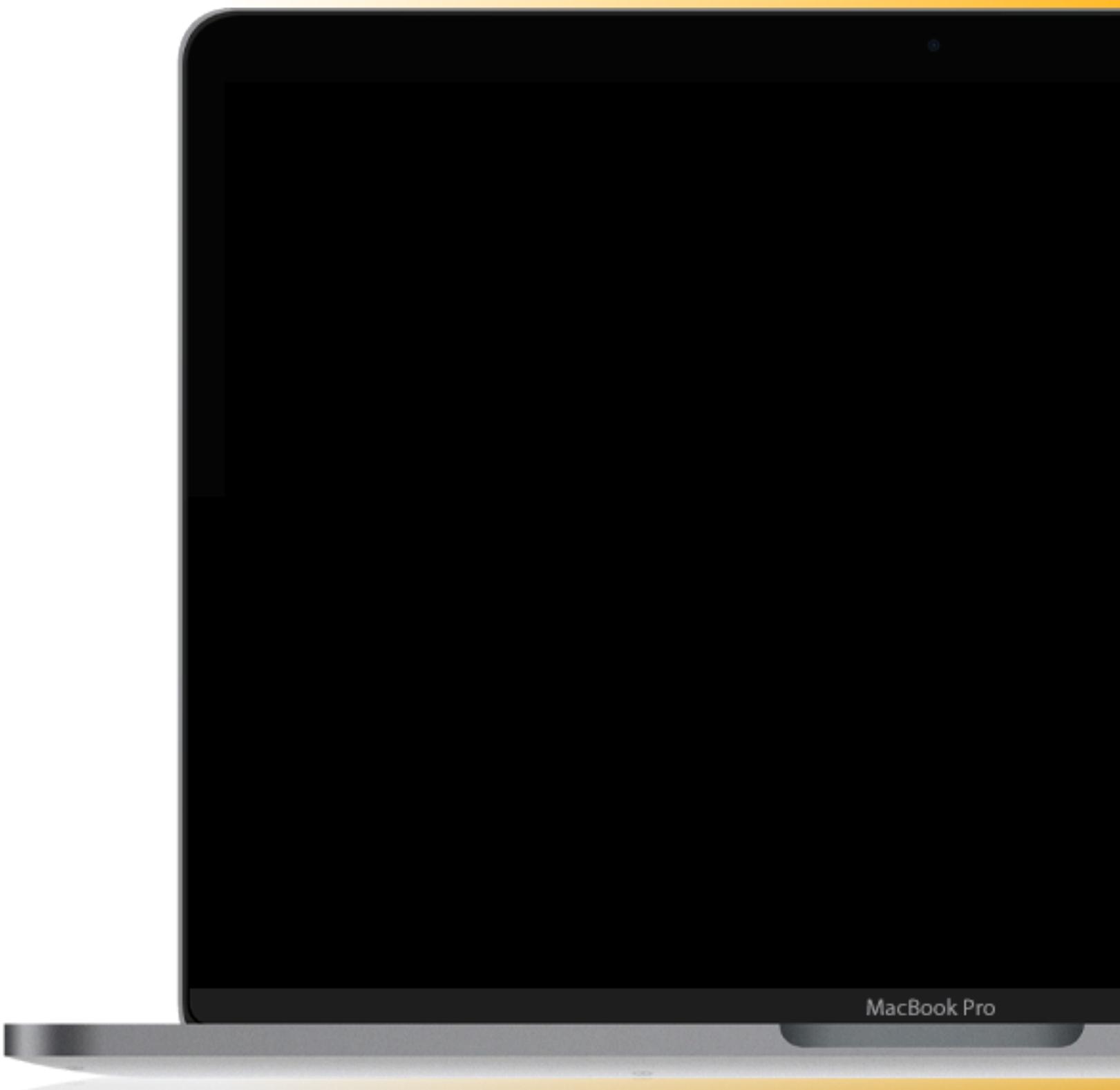




Metodologías de desarrollo de software

Gestor de inventario

Robinson Estrella
Mathias Tapia
Fenix Toapanta





Objetivos General

Desarrollar un sistema de gestión de inventario mediante la aplicación del marco ágil Scrum y la elaboración de historias de usuario, con el fin de automatizar procesos clave, optimizar la administración de recursos y mejorar la toma de decisiones a través de reportes e indicadores precisos.



Objetivos Específicos

- Analizar e identificar los requisitos funcionales y no funcionales mediante el estudio comparativo de sistemas afines.
- Desarrollar y ejecutar casos de prueba estructurados, así como generar reportes de errores para los módulos de control de inventario y registro.
- Diseñar e implementar el sistema utilizando tecnologías apropiadas, como bases de datos relacionales, dentro de un entorno de desarrollo ágil que garantice eficiencia y escalabilidad.

```
ang not in ("en", "fr")):  
eLanguageFrontendImparsing  
iput language could not be determined  
-nputToLanguageModel(inputString,  
    self.model:  
  
nput) # Add new conversation entry  
generateLLMOutput(parsedInput),  
eModel(inputString, inputLanguage  
one or self.model.language != in  
nitalised or has wrong language  
self.loadAILanguageModelFromDat  
l is None or not self.runModel()  
xception("AI language model lo  
None  
tLLMContext(context) # Put pas  
er = self.model.getInputParser  
nputParser.parseInput(inputSt
```

Alcance

- Registro de productos
- Control de movimientos de inventario
- Generación de reportes
- Gestión de usuarios y roles



Resultados Esperados

El desarrollo e implementación del sistema de gestión de inventario generará los siguientes resultados tangibles e intangibles:

- Un sistema funcional con interfaz amigable para la gestión del inventario y ventas de acuerdo a los requisitos del cliente.
- Reducción del tiempo de búsqueda y registro de productos.
- Mejora en la precisión del control de stock.



Ideas a defender

Este proyecto se sustentará en los conocimientos teóricos y prácticos adquiridos en asignaturas aprobadas de la carrera:

- Fundamentos de programación
- Programación Orientada a Objetos
- Bases de Datos
- Metodologías Agiles

Enlaces



Jira

Backlog

Caja Blanca

Confluence

Kanban

Caja Negra

SISTEMA INVENTARIO

PRESENTACION

The image shows a screenshot of a code editor with a dark theme. On the left, the project structure is visible under the 'Project' tab, showing a 'blog' directory with various components like .next, components, context, documentation, graphql, hooks, lib, node_modules, pages, admin, api, public, tests, and utils. It also lists files such as package.json, README.md, esconfig.json, and yarn.lock. Below the project structure is a list of external libraries and scratch files.

The main editor area displays the content of an `_app.tsx` file. The code imports several packages including useEffect from react, Head from next/head, type from next/app, ApolloProvider from @apollo/client, ThemeProvider from @material-ui/core/styles, CssBaseline from @material-ui/core/CssBaseline, Container from @material-ui/core, and useApollo from ../graphql/client. It then imports lightTheme and darkTheme from ../utils/theme and useLocalStorage from ../hooks/useLocalStorage. The component itself uses these imports to set up a NavBar, ApolloClient, and a useEffect hook to handle server-side styles. Finally, it returns a Head component with a title of ECU-DEV and a meta tag for viewport.

```
import { useEffect } from 'react'; 8.23 kB (gzip: 3.33 kB)
import Head from 'next/head';
import type { AppProps } from 'next/app'; 5.11 kB (gzip: 2.16 kB)
import { ApolloProvider } from '@apollo/client'; 123.67 kB (gzip: 33.78 kB)
import { ThemeProvider } from '@material-ui/core/styles'; 2.45 kB (gzip: 1.15 kB)
import CssBaseline from '@material-ui/core/CssBaseline'; 61.61 kB (gzip: 20.02 kB)
import { Container } from '@material-ui/core'; 63.32 kB (gzip: 20.38 kB)
import { useApollo } from '../graphql/client';

import { lightTheme, darkTheme } from '../utils/theme';
import useLocalStorage from '../hooks/useLocalStorage';

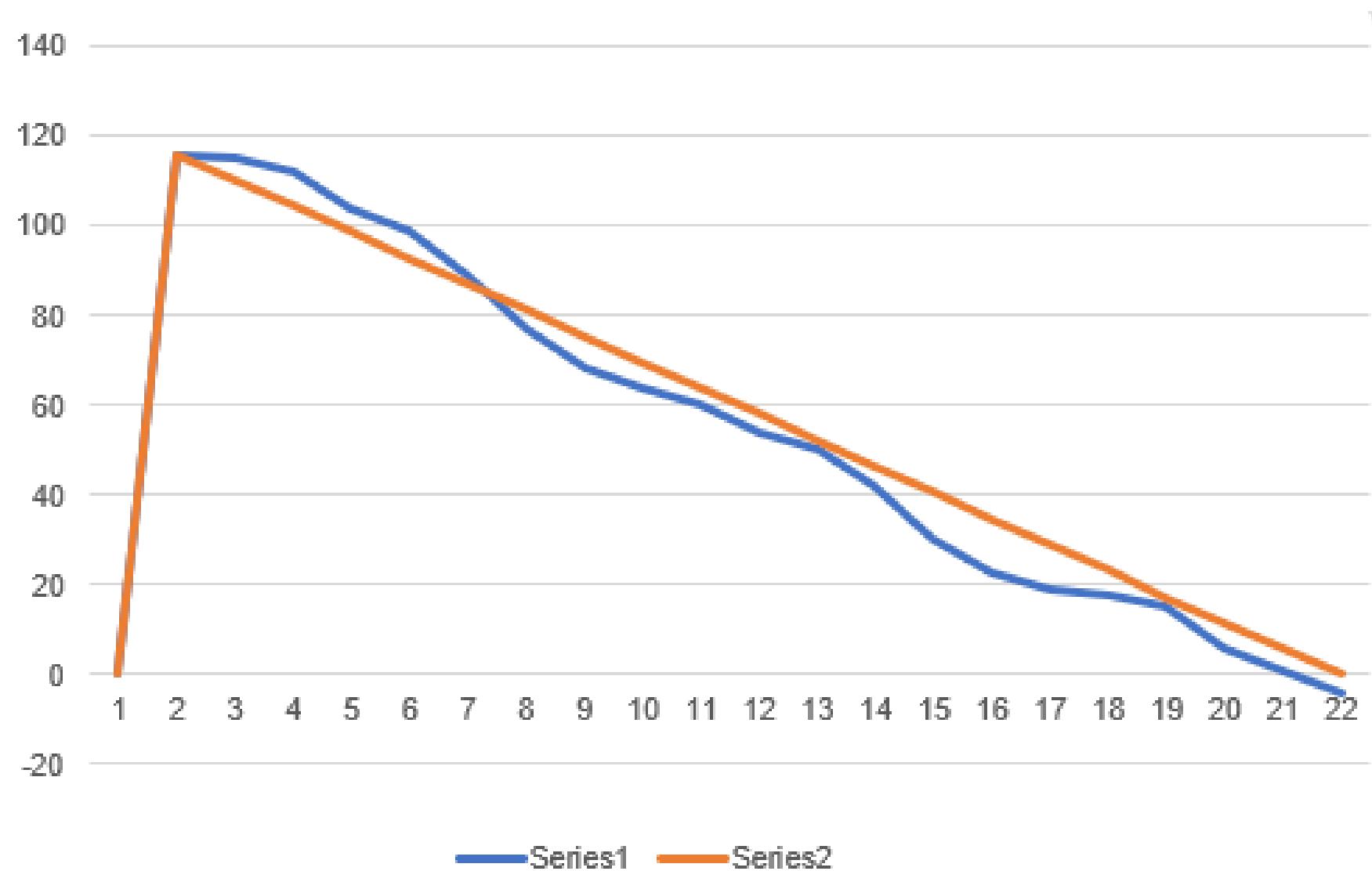
function App({ Component, pageProps }) {
  const [currentTheme, setCurrentTheme] = useState('theme-value', initialValue: 'light');
  const apolloClient = useApollo(pageProps.initialApolloState);

  useEffect(() => {
    const jssServerSideStyles = document.querySelector(selectors: '#jss-server-side');
    if (jssServerSideStyles) {
      jssServerSideStyles.parentNode.removeChild(jssServerSideStyles);
    }
  }, [currentTheme]);
}

return (
  <>
  <Head>
    <title>ECU-DEV</title>
    <meta name="viewport" content="minimum-scale=1, initial-scale=1, width=device-width, height=device-height, shrink-to-fit=no">
  </Head>
  <Component {...pageProps} />
)
```

Grafico Burdonchart

Se puede concluir que al comienzo de los sprints el equipo se retraso, y desde el segundo sprint el equipo siguió el flujo de trabajo establecido





¡Gracias!

