FLUJO DE TRABAJO

Grupo número 3 – Codo a Codo ReactJS

14 DE ABRIL DE 2023

REACT INNOVATORS
Argentina

Organización 3
Miembros 3
Responsabilidades 3
Estandarización 3
Nomenclatura del código 3
Estructura de los proyectos 4
Estilos 4
Extensiones obligatorias 4
Proceso de desarrollo 5
Entorno de desarrollo 5
Gitflow (flujo de desarrollo) 5
¿Cómo se aprobará un pull request? 6
Proyectos 6
Proyecto 01 – Presentación del grupo 6
Proyecto 02 – Todo-List 6
Material 7
Links 7
Links recomendados 7
Extensiones recomendadas 7

Organización

Se utilizará la plataforma Trello[1] para asignar tareas a través de tarjetas utilizando el modelo de gestión Kanban[2], mientras que Discord[3] será el canal de comunicación para establecer reuniones. Se ha seleccionado Discord, debido a que este permite compartir archivos, contener múltiples canales de texto y/o voz, compartir la pantalla sin perder calidad, etc.

I. **Miembros**

Apellido y Nombre	Cargo	Contacto
Hoz Lucas Nahuel	Líder de desarrollo	hozlucas28@gmail.com
Fariña Gladis	Desarrollador/a	lalyfar@hotmail.com
Francisco Carlos	Desarrollador/a	calitoozzy@gmail.com
Acevedo Carlos	Desarrollador/a	carlosacevedo1@yahoo.com
Casasola Marisol	Desarrollador/a	mmarisol.casasola97@gmail.com

II. **Responsabilidades**

- **Líder de desarrollo**: administrar el espacio de trabajo, mediante la plataforma Trello; realizar el seguimiento del proyecto; asignar tareas al equipo; desarrollar funcionalidades; fomentar el desarrollo profesional del equipo.
- **Desarrollador/a**: proponer nuevas tareas; entregar sus tareas asignadas, y en caso de encontrar dificultades documentar los inconvenientes a través de Trello; informar motivo por el cual no pudo continuar con el desarrollo; pedir ayuda al líder de desarrollo y/o al resto del equipo.

Estandarización

En esta sección se declarará una codificación estándar para que el desarrollo de los integrantes sea homogéneo y ordenado. Además, se establecerán las tecnologías y frameworks que se utilizarán durante el desarrollo. En los siguientes puntos se abordarán los temas mencionados:

I. **Nomenclatura del código**: utilizaremos el idioma inglés al declarar variables, debido a que es un estándar en el mundo IT. Además, haremos uso del "lower camel case" para las variables, siendo la nomenclatura menos conflictiva entre los lenguajes de programación. Por último, se hará uso de variables tipo <const> siempre que se pueda.

[1] Trello: aplicación de gestión de proyectos en línea que utiliza un sistema de tarjetas para organizar tareas y colaborar en diversos proyectos.

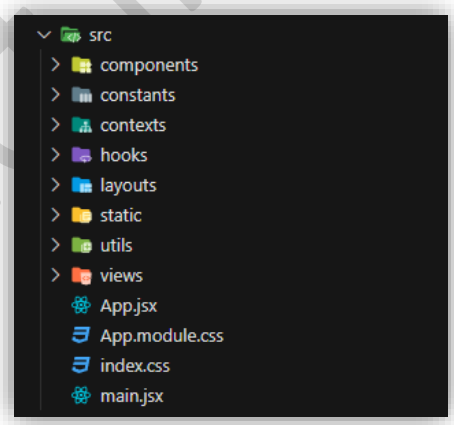
[2] Kanban: método de gestión visual de procesos que se utiliza para optimizar el flujo de trabajo y maximizar la eficiencia en la producción.

[3] Discord: plataforma de comunicación en línea que permite a los usuarios chatear, hacer llamadas de voz y video, y compartir archivos en grupos o canales privados.

II. **Estructura de los proyectos**: los proyectos se basarán en una estructura simple pero eficaz para los proyectos que se desarrollarán, a continuación se mencionarán las carpetas junto con su contenido, tener en cuenta que las mismas estarán ubicadas en la carpeta padre llamada "src".

- "components": como su nombre lo indica, almacena los componentes de nuestra aplicación, junto con sus archivos de estilos.
- "constants": almacena variables globales exportables, como textos y casos de prueba.
- "contexts": almacena los contextos de nuestra aplicación.
- "hooks": posee los custom Hooks.
- "layouts": almacena los componentes de diseño de la aplicación, que definen la estructura general y la apariencia de las páginas.
- "static": guarda todos los archivos multimedia (webp, svg, etc.) que se empleen en los componentes.
- "utils": contiene las funciones exportables que puedan ser empleadas desde cualquier punto de la aplicación. Las mismas cumplirán una función específica, sin importar el contexto de ejecución.
- "views": almacena los componentes de interfaz de usuario que representan la lógica visual de la aplicación.

Imagen ilustrativa de la estructura descrita…



III. **Estilos**: a la hora de emplear estilos utilizaremos módulos CSS, mediante el framework Tailwind CSS. Los módulos CSS nos permiten crear clases dinámicas asegurándonos que dichos estilos declarados se aplicaran únicamente a el componente o etiqueta asociada. Mientras que Tailwind CSS se encargara de realizar el "reset CSS"[4], optimizar la carga de estilos y darle un estilo único a la aplicación mediante "utility classes"[5].

[4] Reset CSS: proceso de estandarización de los elementos de una página web, eliminando los estilos predeterminados de los navegadores y brindando así una base limpia para estilar los mismos.

[5] Utility classes: clases de utilidad que contienen métodos, y funciones comunes que realizan tareas específicas.

IV. **Extensiones obligatorias**: a continuación se listarán las extensiones de carácter obligatorio, ya que las mismas son necesarias para un correcto funcionamiento de las dependencias de desarrollo. Estas aceleraran tu tiempo de desarrollo y te ayudaran a emplear buenas prácticas.

- EsLint: herramienta de análisis de código para JavaScript que ayuda a identificar y corregir errores, y prácticas poco recomendadas.
- Prettier: herramienta de formateo de código que ayuda a mantener un estilo consistente a lo largo de todo el proyecto. Logrando mejorar la legibilidad del código.
- Tailwind CSS IntelliSense: extensión que proporciona asistencia inteligente para escribir código con Tailwind CSS. Ofreciendo autocompletado de clases, información sobre valores y variantes disponibles.

Proceso de desarrollo

I. **Entorno de desarrollo**: al momento de desarrollar se utilizará el entorno de desarrollo ViteJS, debido a que create-react-app ya no es mantenida por el equipo de React y ha quedado descontinuada.

II. **Gitflow (flujo de desarrollo)**

Gitflow es una metodología de trabajo para desarrolladores utilizando Git como herramienta principal. El mismo se basa en las siguientes ramas: “Master”, rama de producción, encargada de almacenar el código estable del proyecto visible para el público; “Development”, que se desprende de “Master”, rama que integra nuevas funcionalidades con el objetivo de realizar pruebas conjuntas en busca de una versión estable del proyecto.

Las nuevas funcionalidades (tareas asignadas) deberán realizarse en un rama propia, extendida de “Development” con la siguiente nomenclatura: feature/<FUNCIONALIDAD>. Una vez que la funcionalidad se haya desarrollado y cumpla con los requisitos esperados, se realizara un pull request[6] para aprobar su merge[7] a la rama “Development”.

Cuando se llegue a una versión estable del proyecto en la rama “Development”, se creará una rama llamada “release”, extendida de “Development”, donde se efectuará documentación, solución de errores, y tareas orientadas a la publicación. Una vez concluido el desarrollo de “release” se realizará su merge a “Master” y reiniciar así el flujo.

A continuación se resume el flujo de Gitflow:

1º. Se crea una rama “Development” a partir de la rama “Master”.

2º. Se crean ramas “feature” a partir de la rama “Development”.

3º. Cuando se termina una rama “feature”, se la fusionara con la rama “Development”.

[6] Pull request: solicitudes que los desarrolladores hacen para fusionar sus cambios en un proyecto, permitiéndole a otros revisar y discutir los cambios antes de realizar el merge.

[7] Merge: proceso de combinar los cambios propuestos en un pull request con una rama del proyecto.

4º. Cuando la rama “Development” sea estable se creará una rama “release” a partir de “Development”.

5º. Cuando la rama “release” este lista, se la fusionara con las ramas “Development” y “Master”.

6º. Si se detecta un problema en “Master”, se creará una rama “hotfix” a partir de “Master” para solucionar el problema.

7º. Una vez solucionados los problemas en la rama “hotfix”, esta se fusionará tanto en “Development” como en “Master”.

III. **¿Cómo se aprobará un pull request?**

Para que un pull request sea aprobado y se realice el merge a una rama, dicho PR (pull request) deberá tener el visto bueno de al menos el 50% de los desarrolladores del equipo y del líder, es decir, 2 de 4 desarrolladores y el líder del equipo deberán dar el OK al PR, para que el merge sea ejecutado. No se esperará la aprobación del 100% de la plantilla debido a que este proceso afectaría negativamente los tiempos de desarrollo.

Proyectos

I. **Proyecto 01 – Presentación del grupo**: se deberá presentar al grupo en una tarjeta, a través de una aplicación web hecha con ReactJS. La información del grupo deberá ser extraída de un archivo <.json>, que contendrá todos los grupos del programa. Dicha aplicación debe cumplir con los siguientes puntos.

- Mostrar el ID del grupo.
- Mostrar el nombre del líder del grupo.
- Mostrar los nombres de los desarrolladores.
- Mostrar el nombre del grupo, elegido por los integrantes.

II. **Proyecto 02 – Todo-List**: se deberá elaborar un aplicación web que almacene un listado de tareas pendientes, pudiendo agregar más de estas al listado existente. Dicha aplicación debe cumplir con los siguientes puntos.

- Mostrar el listado de tareas pendientes.
- Las tareas completadas deberán poder ser eliminadas definitivamente del listado.
- Se deberá poder modificar los títulos y descripciones de las tareas una vez listadas.
- Al cliquear sobre una tarea, marcar como completa la misma, tachando y cambiando de color su contenido.
- Mostrar un formulario para agregar una nueva tarea al listado existente, dicha tarea deberá tener un título y una descripción.

Material

I. Links:

- Espacio de trabajo - Trello
- Canal de comunicación para reuniones - Discord
- Repositorio del proyecto 01 – Presentación del grupo
- Repositorio del proyecto 02 – Todo-List

II. Links recomendados:

- Anotaciones - Git
- Anotaciones - JavaScript
- Anotaciones - ReactJS (básico)
- Anotaciones – ReactJS (Hooks)
- Flujo de trabajo de Gitflow

III. Extensiones recomendadas:

- Error Lens: resalta visualmente los errores, advertencias y otros problemas en el código mientras se escribe, mostrando información adicional y sugerencias de corrección, facilitando la identificación y solución de problemas.
- Spell Right: proporciona corrección ortográfica en tiempo real, utilizando el diccionario del usuario en cuestión.
- Material Icon Theme: cambia los iconos predeterminados de los archivos y carpetas del explorador de archivos con iconos personalizados, facilitando la identificación visual y la navegación.
- ES7+ React/Redux/React-Native snippets: proporciona un conjunto de atajos de teclado (snippets) para escribir código React, Redux y React-Native de manera más rápida y eficiente, ofreciendo plantillas de código para componentes, entre otras instrucciones.