(v0.0.3 - 03-08-2024)

Team Lead: Milton Loaiza

Prueba de desempeño TypeScript

Propósito:

Como equipo de formación Riwi, queremos evaluar las habilidades de desarrollo en el lenguaje de programación TypeScript de un aspirante a Junior developer, específicamente en el proceso de implementación de soluciones a un problema o necesidad con este lenguaje en el ambiente front-end. Esto nos permitirá determinar si el candidato posee los conocimientos técnicos y la capacidad de resolución de problemas necesarios para contribuir efectivamente en proyectos reales y en entornos laborales.

Metodología:

La prueba técnica se estructurará como una historia de usuario Scrum con una duración estimada de 8 horas. Esta metodología ágil permite simular un entorno de trabajo realista, donde los desarrolladores reciben tareas con plazos definidos y deben gestionar su tiempo y recursos de manera eficiente.

No se permitirá el uso de IA, ni consultas externas, verificando así la apropiación real de conocimientos sobre los elementos vitales del lenguaje de programación que no se deberían delegar a sistemas externos. Las herramientas permitidas son el editor de código, Postman, el navegador para probar la plataforma y validar documentación de las API, la terminal para uso de Git y demás. Otras herramientas que se requieran utilizar deberán ser bajo aprobación del TL.

La evaluación se centrará en la capacidad del candidato para aplicar los conocimientos teóricos en la práctica, resolver problemas de manera creativa y demostrar un dominio sólido de las tecnologías y conceptos fundamentales de TypeScript.

Esta metodología no solo evalúa las habilidades técnicas del candidato, sino también su capacidad de adaptación y gestión del tiempo, cualidades esenciales para un Junior developer exitoso.

Historia de usuario (Enunciado):

¡Bienvenido Coder a tu prueba de desempeño de (TypeScript)!

- **Las reglas:**
    - Tienes 8 horas máximo para la culminación de la prueba. (6am a 2pm) con 3 descansos (8:00 am a 8:20 am, 10:00 am a 10:20 am y 12:00 m a 12:20 m)
    - Debes tener la capacidad de explicar tu código.
    - Debes realizar la prueba de forma individual.
    - Se deben cumplir los criterios de aceptación de la HU.

- **Las metodologías:**
    - Trabajarás en una historia de usuario similar a un entorno de trabajo real.
    - Demuestra tus conocimientos técnicos y tu capacidad para resolver problemas.
    - ¡Buena suerte y manos a la obra!

- **Historia de usuario:**

  Empresa: Data Leaker Gates

  Contexto: Empresa dedicada a la gestión de información de calidad e información segura para los usuarios de foros, chats y demás a nivel Colombia, tiene la necesidad de implementar una funcionalidad que permita validar una entrada de texto para que cumpla con los estándares determinados como prohibir palabras obscenas o reservadas, buen manejo ortográfico y no permitir agregar entradas que no cumplan esa calidad.

  Tarea asignada:
  Como desarrollador junior de una plataforma de gestión de información de calidad e información segura para los usuarios, quiero implementar las siguientes funcionalidades:

  - Un front-end web realizado con TypeScript, utilizando las mejores prácticas de desarrollo de software como código limpio, programación orientada a objetos, patrón de arquitectura Modelo Vista, Controlador u otro más avanzado que puedan justificar.
  - Uso adecuado de HTML, CSS y su implementación en lo posible como componentes.
  - Vista de registro que permita crear un nuevo usuario con solo email y password, consumir API de gestor de posts para crear el usuario y dar buen manejo de errores o mensajes de no autenticado.
  - Vista login que permita ingresar con email y password del usuario que va a trabajar, validándolo con la API y si está autorizado almacenarlo en el session storage y llevarlo a la vista home.
  - Vista home que muestre un listado de post creados por ese usuario en donde se observe datos relevantes como: fecha de creación,

fecha estimada de publicación, plataforma (Twitter, Facebook, etc.), porcentaje de calidad de la publicación con un color que indique si cumple (verde) o si no cumple (rojo) con los criterios de calidad, el título de la publicación, además de botones o elementos de acción para editar, ver en detalle y eliminar. El Home debe tener la opción para crear un nuevo post. El post se debe ir almacenando en localStorage hasta que se envíe al back-end, para que si por accidente se sale no se pierda la información ingresada.

- Vista creación/edición de post, esta vista debe permitir crear o actualizar el post según los datos previstos en la API de gestión. Además, cuando el usuario finalice la creación, o actualización, antes de irlo a crear en la API de gestión, se debe validar las palabras excluidas que deben estar en una lista quemada, por ejemplo groserías o palabras prohibidas, reservadas. Se debe detectar la presencia de palabras que afectarán el porcentaje de calidad del post. Además, se debe consumir la API de corrección ortográfica y reportar dichas palabras también como de baja calidad.
- El cálculo del cumplimiento de la calidad del post es: si más del 5% de las palabras tienen inconvenientes (palabras excluidas o errores ortográficos) debe marcarse los errores y reportar el porcentaje de calidad y no dejar grabar en la API solo en local.
- En la vista de detalle del post, las palabras prohibidas deben marcarse al igual que las palabras con errores ortográficos, deben hacerlo de forma que se pueda diferenciar entre errores ortográficos y palabras prohibidas, para que el usuario pueda editar lo incorrecto.
- En todo momento se debe validar a través de pop-ups, alerts o Toast o cualquier otra técnica apropiada del front-end para notificar al usuario en caso de errores o éxitos en los procesos.

- En todo momento se debe mostrar cuando el sistema está cargando o a la espera, pueden usar algunos de los métodos convencionales como círculos de carga, entre otros.
- Se espera un front-end estético y bien ordenado.

Plus (No obligatorio)

- Que al crear el post me permita subir una imagen para el post a través de Cloudinary y enviar la URL a la API de gestión.
- Desplegar el frontend en la nube en cualquier proveedor.
- Implementar al menos una prueba unitaria con Jest.

Tiempo: 8 horas
Se recomienda distribuir el tiempo adecuadamente.

Criterios de Aceptación:

- Uso de APIs:
  - Utilizar las estructuras más adecuadas de TypeScript para la recepción y envío de datos a la API (interface, class, etc.)
  - Validar métodos y end-points para el uso correcto de la API.

- Control adecuado de errores:
  Validar las respuestas de la API y mostrar efectivamente mensajes al usuario, además controlar posibles errores y avisar al usuario.

- Documentación:

- Incluir un archivo README.md que explique cómo configurar y ejecutar (Levantar) el proyecto.

- Git-flow:
  - Utilizar Git-flow para la gestión de ramas y versiones del proyecto.
  - Asegurarse de que los commits sean descriptivos y las ramas sigan la convención de Gitflow.

- Uso de sintaxis adecuada de TypeScript:
  - Asegurarse de que en lo máximo posible todo este tipado.
  - Utilizar herramientas provistas por el lenguaje de programación como el map, foreach, filter, entre otras cuando sea adecuado.

Recursos:

- API gestión de posts:
  Host: https://api-posts.codificando.xyz
  Documentación: https://github.com/Richard-01/api-posts
  Documentación de endpoints en postman:
  https://github.com/Richard-01/api-posts/blob/master/PostmanCollections/REST%20API-%20CRUD%20-%20POSTS.postman_collection.json

- API de corrección y validación ortográfica:
  Host: https://api.languagetool.org
  Método: POST

Endpoint: /v2/check
Query params: ?text=your text to analize&language=es
Documentación: https://languagetool.org/http-api/

Entregables:

- Código fuente del proyecto en un repositorio GitHub. Adjuntar url del repositorio y el código en ZIP en Moodle. (solo se revisan ramas con commits que máximo tengan la fecha y hora de finalización de la prueba).

Notas Adicionales:

Se valorará la claridad y organización del código, así como la correcta implementación de las herramientas y prácticas solicitadas.