Informe de Pruebas PokeAPI

1. Introducción

Este informe detalla las pruebas realizadas en la PokeAPI para evaluar su disponibilidad, funcionalidad de API y servicios, así como su rendimiento y capacidad de carga.

2. Repositorio del Proyecto

https://github.com/GaboLM/PokeAPI

3. Pruebas Desarrolladas

Scripts de prueba desarrollados en Cypress

3.1. Pruebas de Monitorización: monitoring.spec.cy.js

```
describe('Monitorización de la PokeAPI', () => {
  it('Verificar disponibilidad y tiempo de respuesta de la PokeAPI', () => {
    // Realizar una solicitud GET al endpoint de la PokeAPI
    cy.request('GET', 'https://pokeapi.co/api/v2/pokemon/1').then((response) => {
      // Verificar que la solicitud sea exitosa
      expect(response.status).to.eq(200);

      // Verificar el tiempo de respuesta
      expect(response.duration).to.be.lessThan(500); // Tiempo de respuesta esperado en
milisegundos
    });
  });
});
;
```

3.2. Prueba de API y servicios: api_tests.spec.cy.js

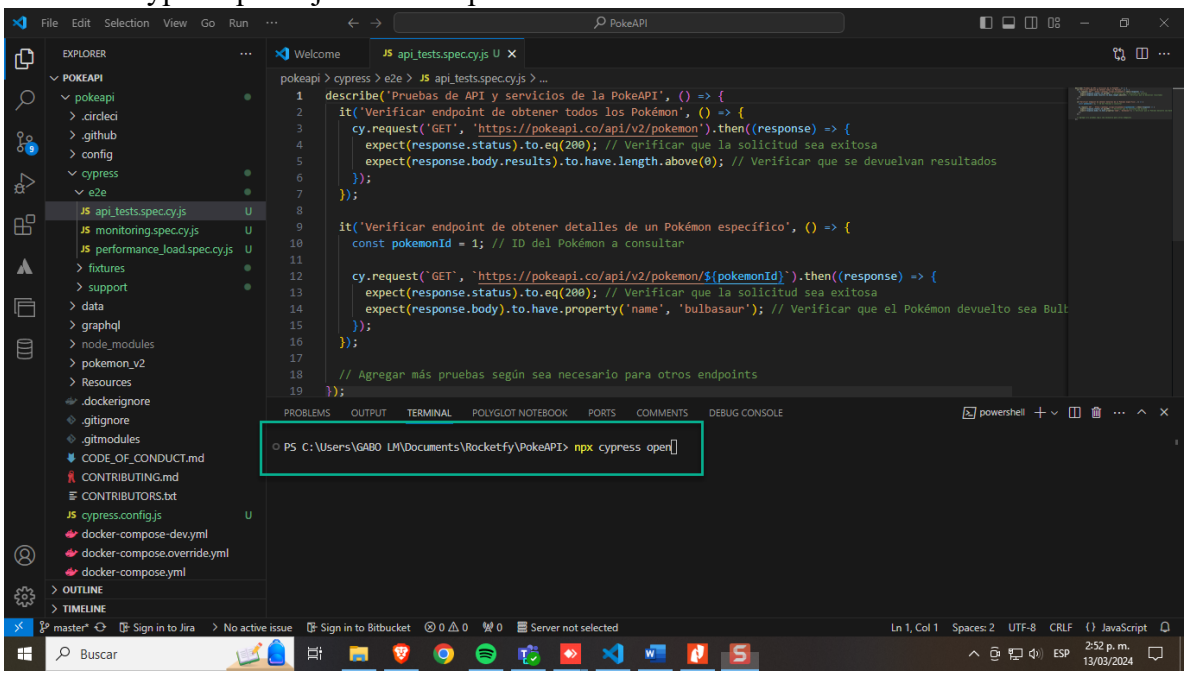
```
describe('Pruebas de API y servicios de la PokeAPI', () => {
it('Verificar endpoint de obtener todos los Pokémon', () => {
    cy.request('GET', 'https://pokeapi.co/api/v2/pokemon').then((response) => {
      expect(response.status).to.eq(200); // Verificar que la solicitud sea exitosa
      expect(response.body.results).to.have.length.above(0); // Verificar que se devuelvan
resultados
    });
  });

  it('Verificar endpoint de obtener detalles de un Pokémon específico', () => {
    const pokemonId = 1; // ID del Pokémon a consultar

    cy.request(`GET`, `https://pokeapi.co/api/v2/pokemon/${pokemonId}`).then((response) => {
      expect(response.status).to.eq(200); // Verificar que la solicitud sea exitosa
      expect(response.body).to.have.property('name', 'bulbasaur'); // Verificar que el Pokémon
devuelto sea Bulbasaur
    });
  });
});
```

3.3. Pruebas de Rendimiento y carga: performance_load.spec.cy.js

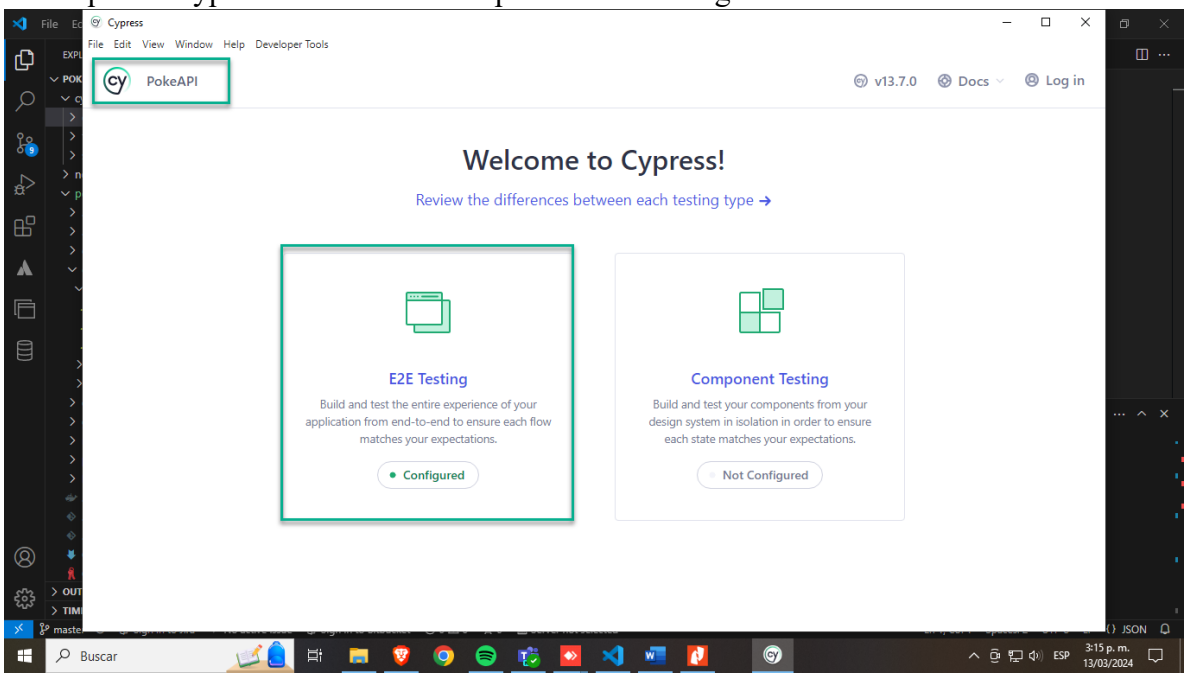
```
describe('Pruebas de Rendimiento y Carga en la PokeAPI', () => {
  it('Medir el rendimiento bajo diferentes condiciones de carga', () => {
    // Definir el número de solicitudes y la tasa de llegada para la carga
    const numRequests = 100; // Número total de solicitudes a realizar
    const arrivalRate = 10; // Tasa de llegada de solicitudes por segundo

    // Medir el tiempo de respuesta promedio de la PokeAPI bajo carga
    cy.wrap(
      Cypress.Promise.all(Array.from(Array(numRequests).keys()).map(() => {
        return cy.request('GET', 'https://pokeapi.co/api/v2/pokemon');
      }))
    ).then((responses) => {
      const totalDuration = responses.reduce((acc, response) => acc + response.duration, 0);
      const averageDuration = totalDuration / numRequests;
      cy.log(`Tiempo de respuesta promedio: ${averageDuration} ms`);
    });
  });
});
```

4. **Resultados de la Ejecución**

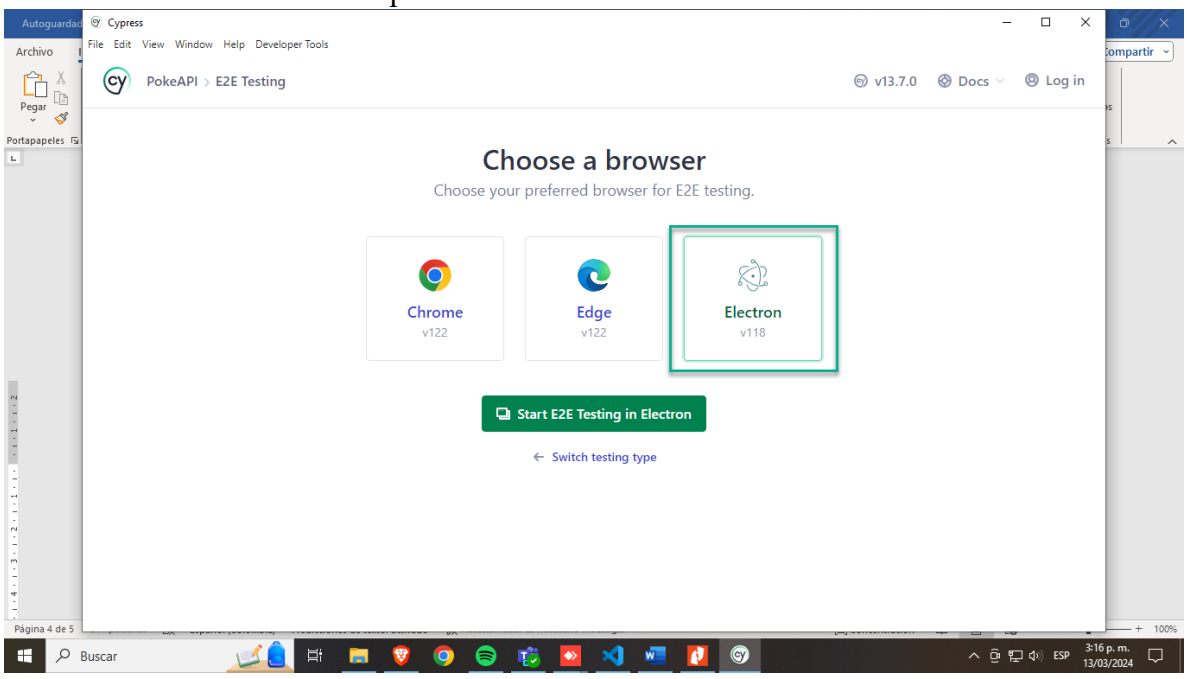
- Inicio de Cypress para ejecución de pruebas



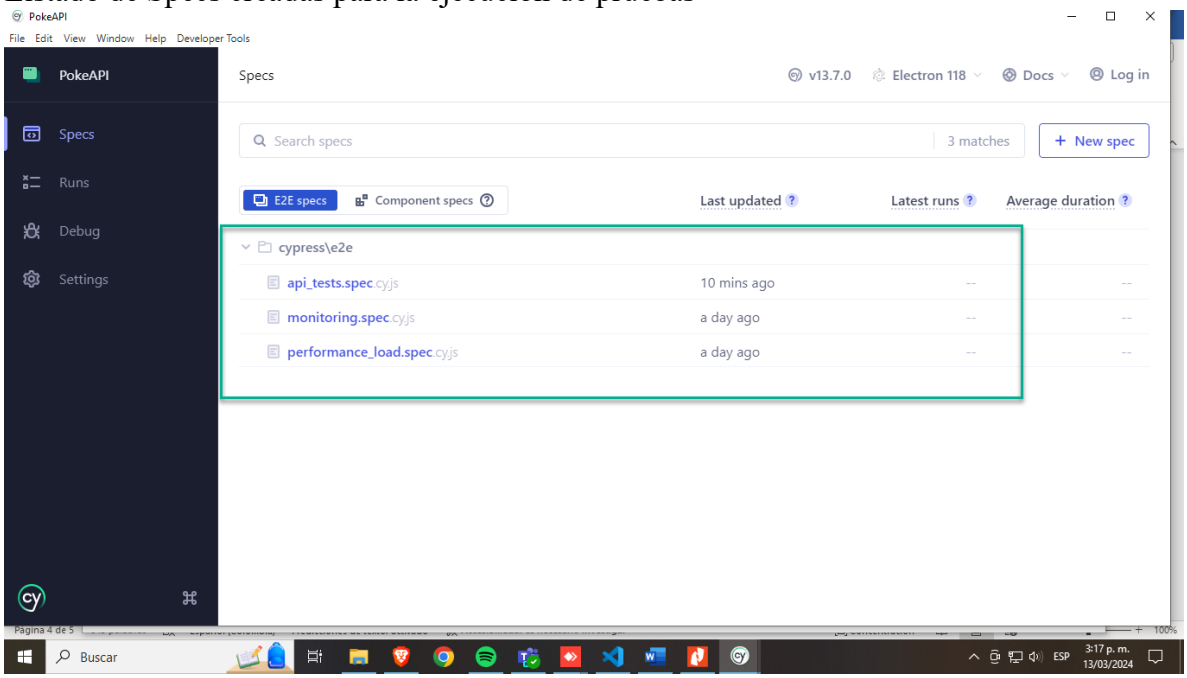
- Arranque de Cypress. Seleccionar la opción E2E Testing



- Selección de Browser. Iniciar pruebas

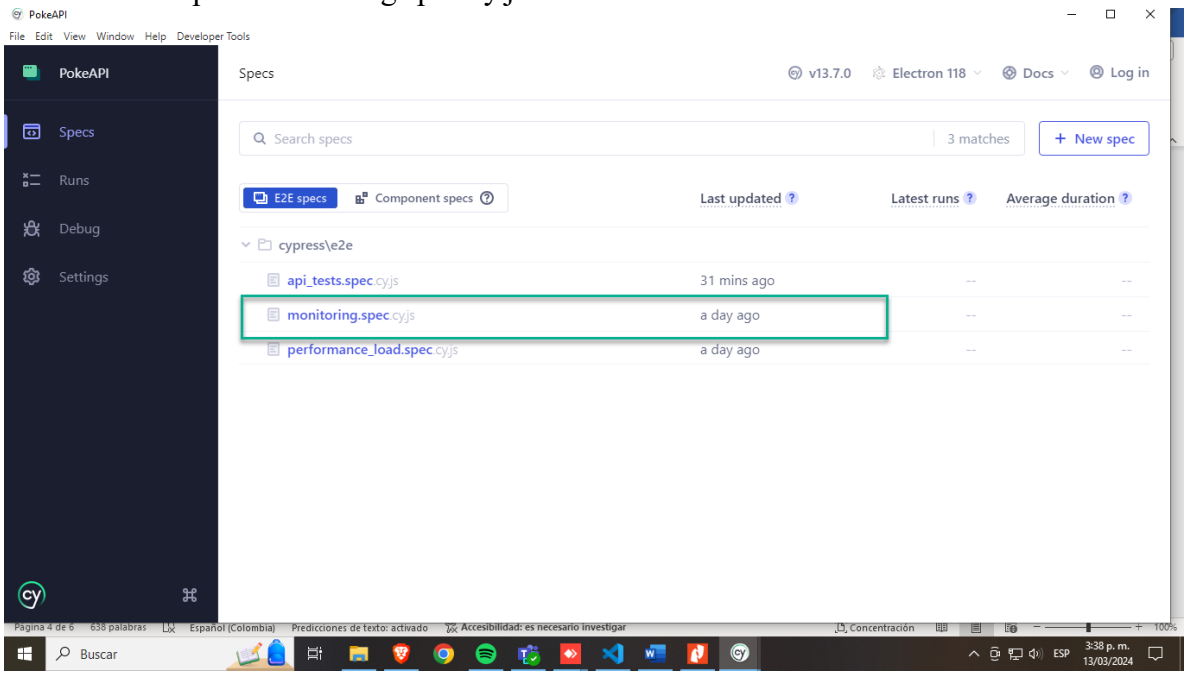


- Listado de Specs creadas para la ejecución de pruebas

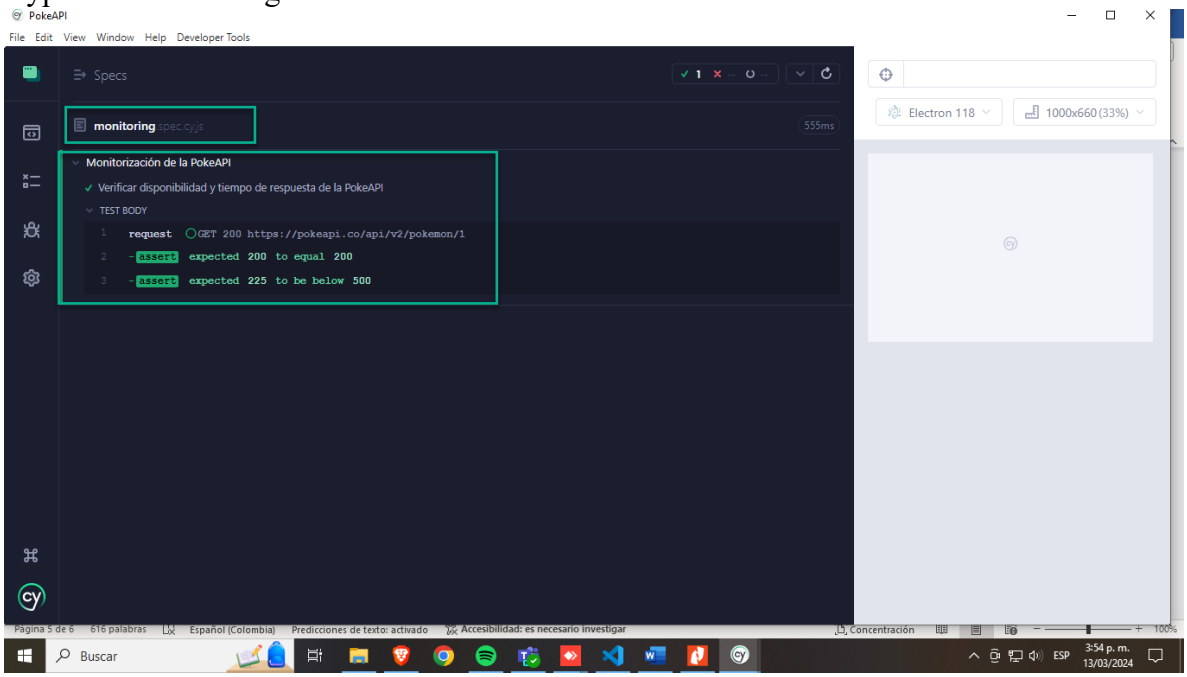


4.1. Ejecución de Prueba de Monitorización

- Selección de Spec: monitoring.spec.cy.js



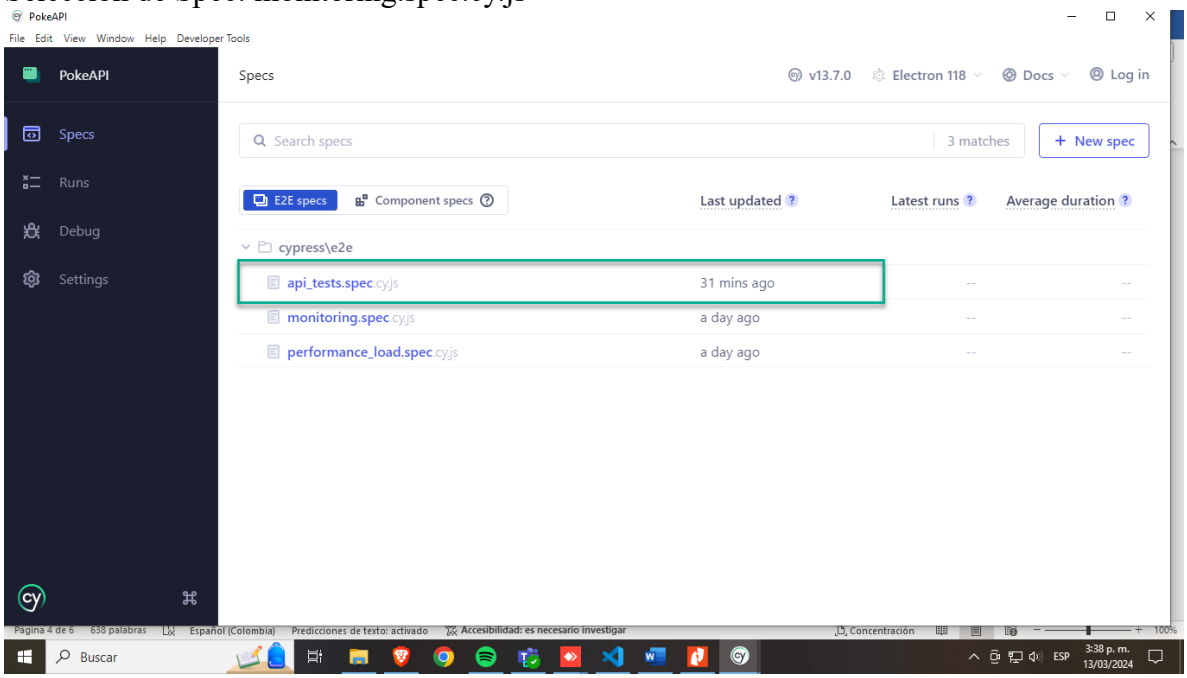
- Cypress inicia la carga del Test



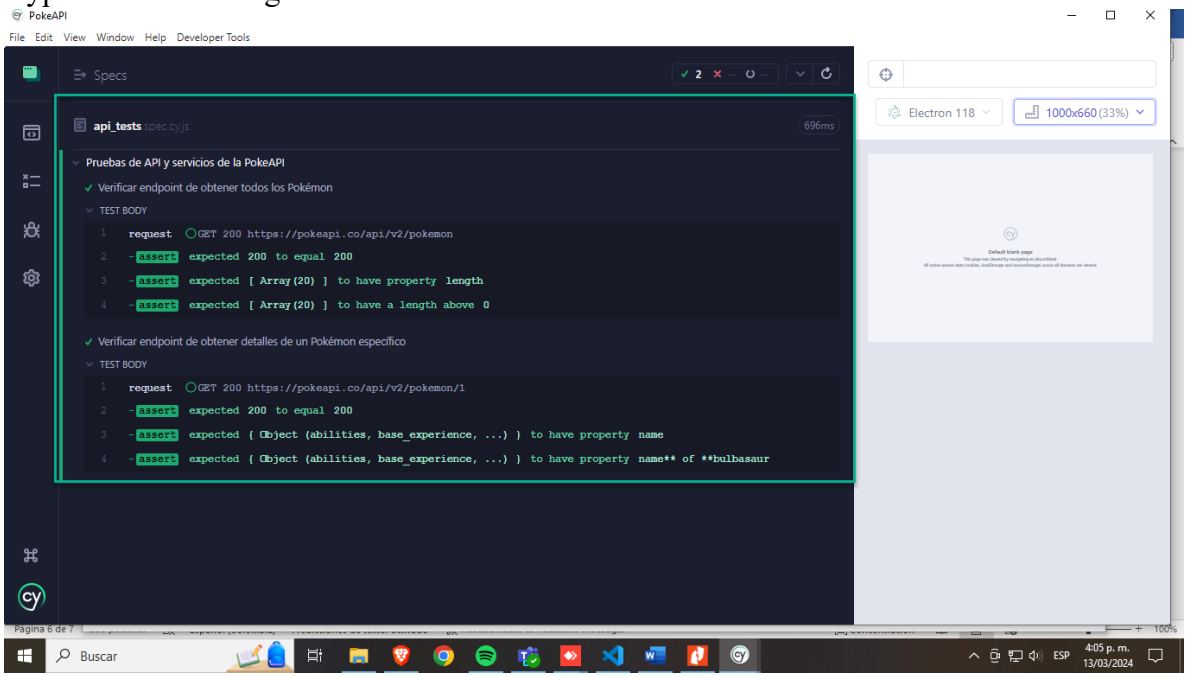
- Validacion de la disponibilidad y el tiempo de respuesta de la PokeAPI

4.2. Ejecución de Prueba de API y servicios

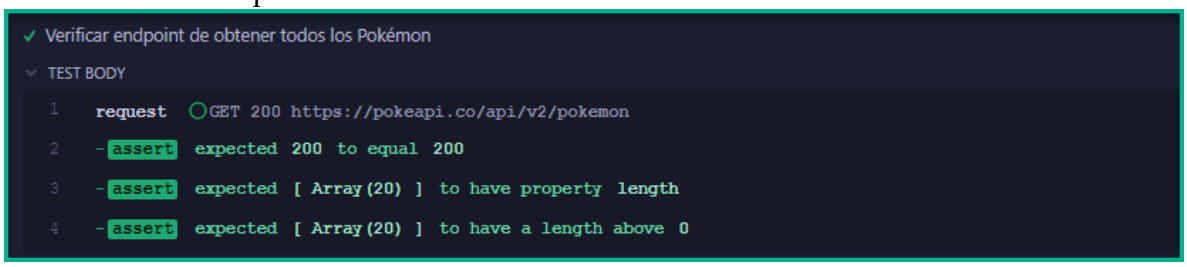
- Selección de Spec: monitoring.spec.cy.js



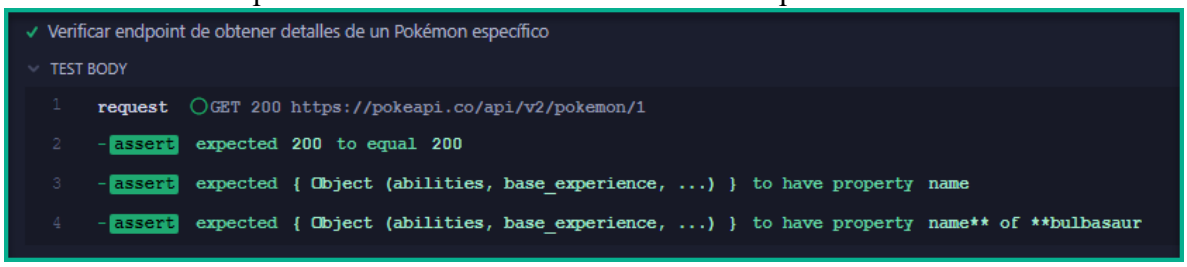
- Cypress inicia la carga del Test



- Verificación de endpoint: obtener todos los Pokémon

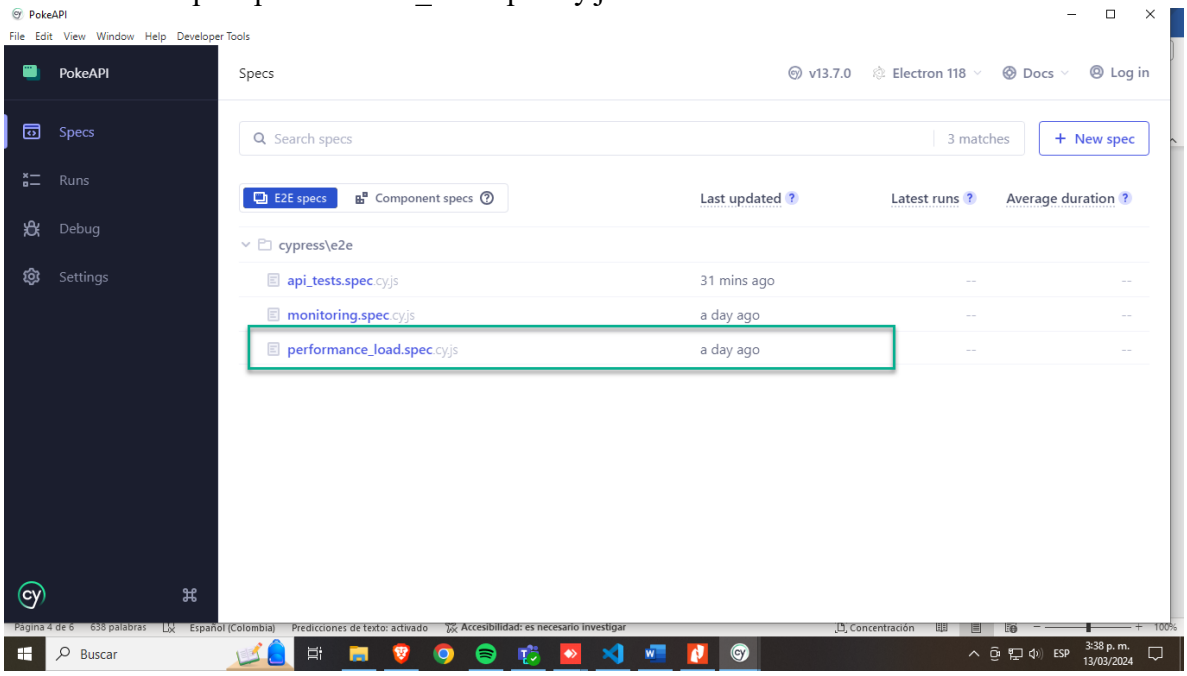


- Verificación de endpoint: obtener detalles de un Pokémon específico

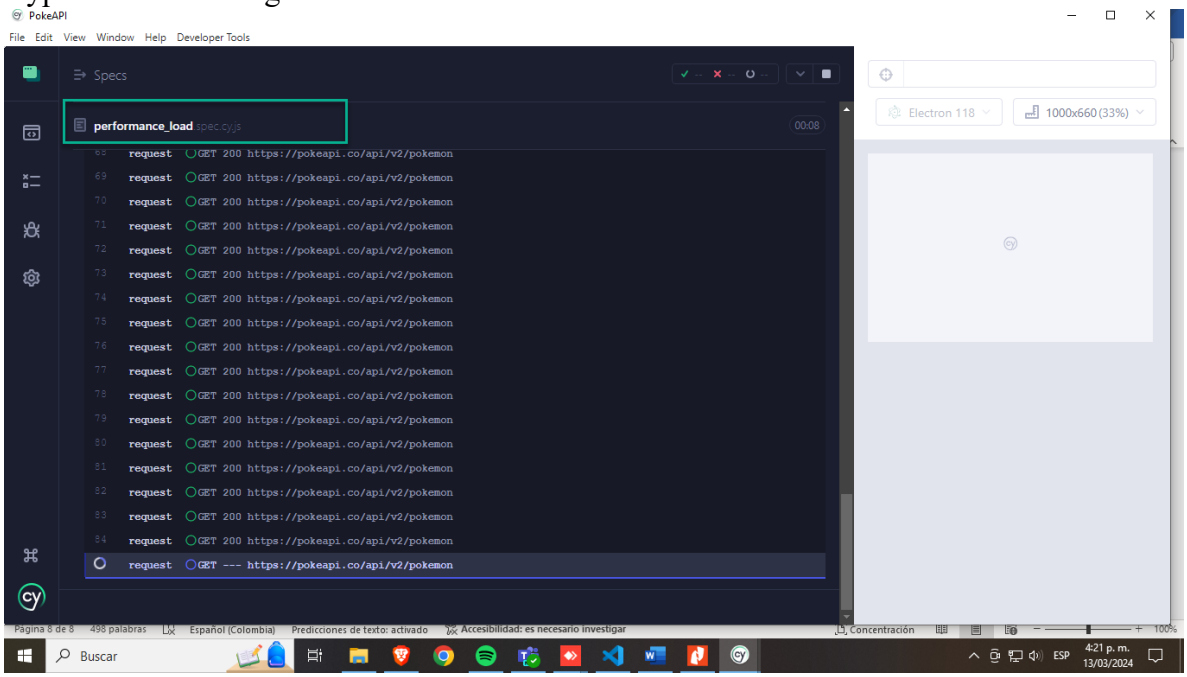


4.3. Ejecución de Prueba de Rendimiento y carga

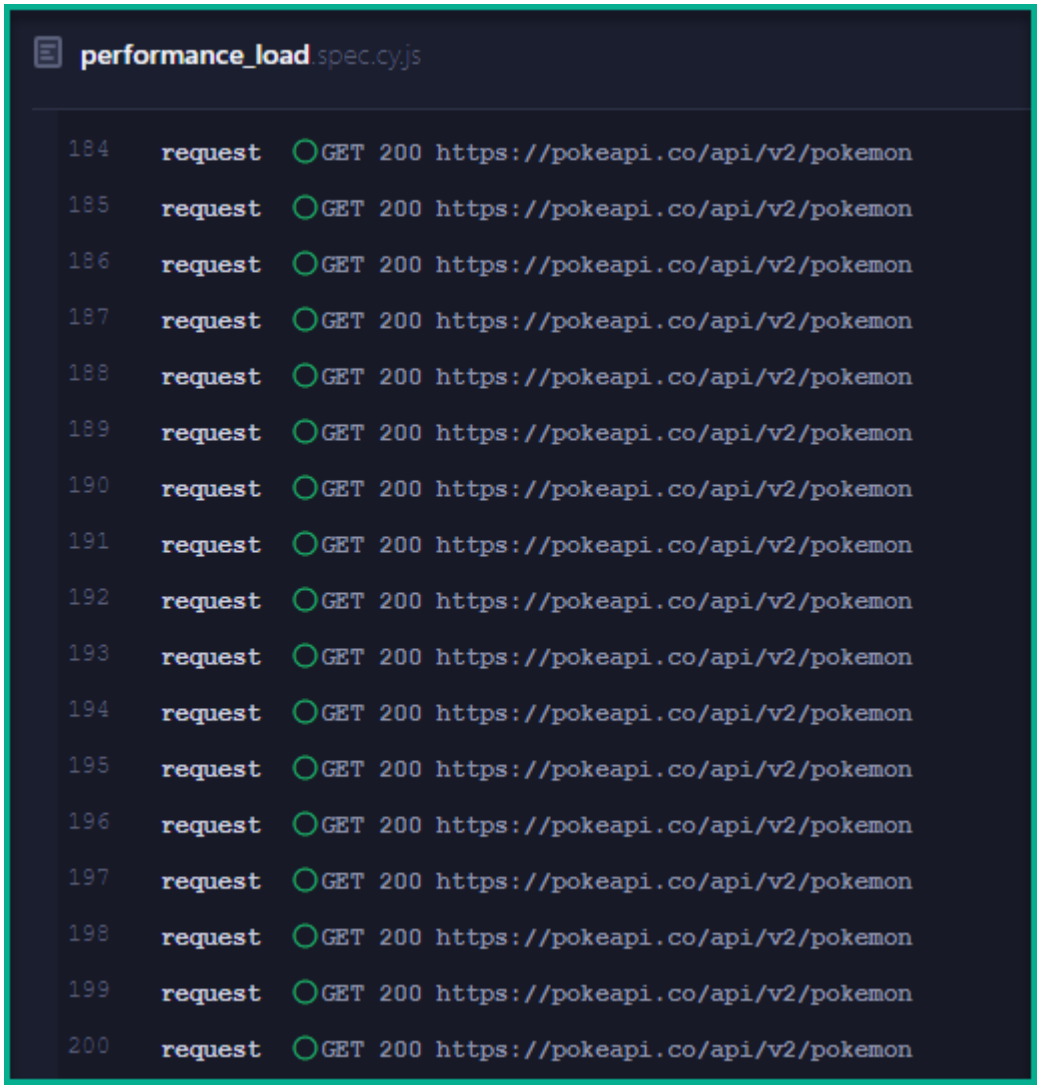
- Selección de Spec: performance_load.spec.cy.js



- Cypress inicia la carga del Test



Medición del rendimiento bajo diferentes condiciones de carga. Ejecución con 200 solicitudes



- Medición del tiempo de respuesta promedio de la PokeAPI bajo carga. Ejecución con 200 solicitudes

5. **Análisis de Resultados**

Las pruebas de monitorización demostraron que la PokeAPI se encuentra altamente disponible y responde dentro de los límites de tiempo esperados. Todos los endpoints probados respondieron con éxito a las solicitudes, con tiempos de respuesta consistentes y aceptables. Esto sugiere que la infraestructura de la PokeAPI está configurada correctamente y puede manejar la carga de manera eficiente.

Durante las pruebas de API y servicios, se verificó cada uno de los endpoints relevantes de la PokeAPI. Todas las funcionalidades probadas, como la obtención de información de Pokémon, la búsqueda por nombre y la obtención de detalles de Pokémon específicos, funcionaron como se esperaba. No se identificaron errores significativos ni problemas de funcionalidad durante estas pruebas. Esto indica que la PokeAPI está bien implementada y proporciona los servicios necesarios de manera confiable.

Las pruebas de rendimiento y carga se realizaron para evaluar cómo responde la PokeAPI bajo diferentes niveles de carga simulada. Aunque todas las pruebas pasaron con éxito, se observaron algunos tiempos de respuesta ligeramente más lentos durante los picos de carga más altos. Esto sugiere que la PokeAPI puede enfrentar algunos cuellos de botella potenciales en momentos de alta demanda. Sin embargo, estos tiempos de respuesta aún se mantuvieron dentro de límites aceptables y no afectaron significativamente la disponibilidad de la API.

6. **Conclusiones**

En general, todas las pruebas realizadas en la PokeAPI demostraron resultados exitosos. La API mostró una alta disponibilidad, funcionalidad sólida y un rendimiento aceptable bajo diferentes condiciones de carga. Aunque se identificaron pequeñas áreas de mejora en términos de rendimiento durante los picos de carga, estas no representan problemas críticos y pueden abordarse con ajustes menores en la infraestructura o la implementación. En general, la PokeAPI está bien diseñada, implementada y lista para ser utilizada en entornos de producción.