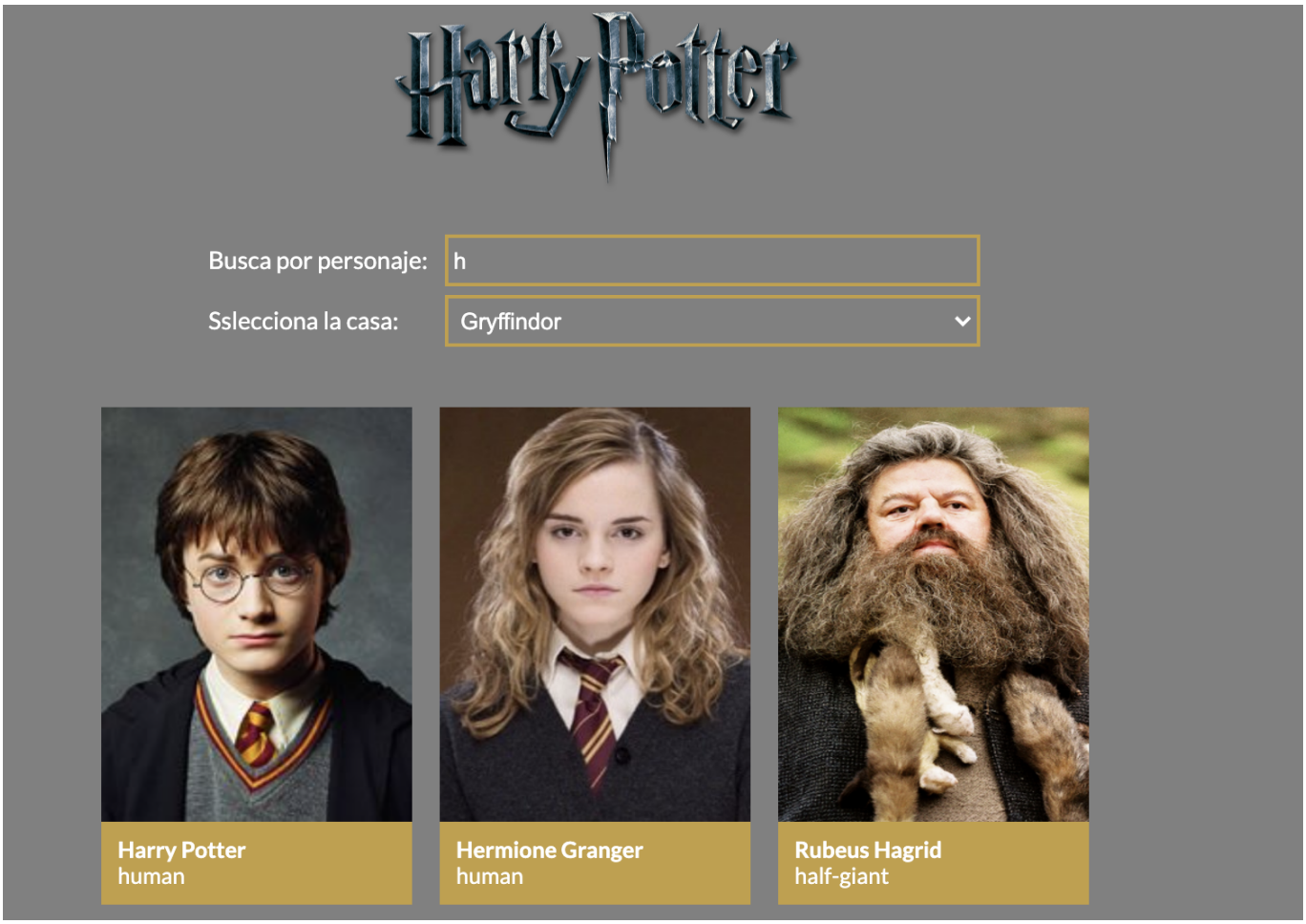
Módulo 3: Ejercicio de evaluación final

Antes de empezar a realizar el ejercicio, hay que crear un nuevo repositorio desde GitHub Classroom usando este enlace. Una vez creado, hay que clonarlo en nuestro ordenador y en la carpeta creada empezaremos a trabajar en el ejercicio.

El ejercicio consiste en desarrollar una página web con un listado de personajes de *Harry Potter*, que podemos filtrar por el nombre del personaje. Vamos a usar React para realizarlo.



Vamos de definir las distintas partes del ejercicio:

1. Listado de personajes

En primer lugar, vamos a realizar una web con el listado de personajes de *Harry Potter*. Para eso, vamos a utilizar el servicio de https://hp-api.onrender.com/ que nos devuelve información sobre los personajes de Harry Potter filtrados por la casa a la que pertenece. Sobre cada uno, vamos a pintar al menos:

- Foto
- Nombre
- Especie

Puedes ingresar a su página oficial del servicio https://hp-api.onrender.com/ para consultar más información.

Imágenes

Algunas de los personajes que devuelve el API no tienen imagen (como, por ejemplo, los Dursley: Vernon Dursley, Petunia Dursley, ... o James Potter). En ese caso hay que mostrar una imagen de relleno. Podemos crear una imagen de relleno con el servicio de **placehold.co** donde en la propia URL indicamos el tamaño, colores, texto: https://placehold.co/210x295/ffffff/666666/?format=svg&text=Harry+Potter, o puedes usar tu creatividad...

2. Filtrado de personajes

Ahora que ya tenemos el listado de personajes en pantalla, la segunda parte consiste en poder buscarlos por nombre. Para eso, añadimos un `input` a la interfaz, de forma que al ir escribiendo un nombre queden en la interfaz solo los personajes cuyo nombre contiene las letras escritas. En el pantallazo de arriba, al escribir 'h' aparecen personajes cuyo nombre completo contiene esas letras en ese orden.

> **Nota:** en principio no es necesario tener en cuenta si las letras están en mayúscula / minúscula para la búsqueda, pero si queréis añadir esta mejora pues genial.

3. Filtrado por casa

Ahora que ya tenemos el listado de personajes en pantalla, y filtrado por nombre la siguiente parte consiste en poder buscarlos por casa. Para eso, añadimos un `select` a la interfaz, de forma que al seleccionar una casa queden en la interfaz solo los personajes cuya casa es la seleccionada.

> **Nota:** Por defecto, cuando carga la página debe aparecer la casa `gryffindor`.

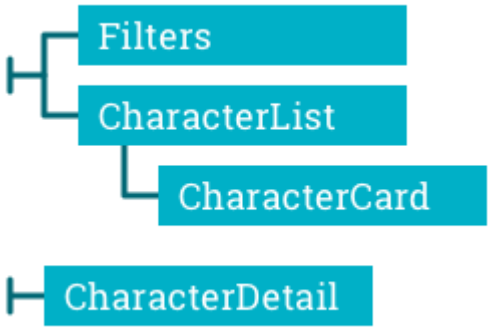
> **2ª nota:** Fíjate que la propia API permite filtrar los resultados por casa.

4. Componentes del listado de personajes

El listado debe tener los siguientes componentes como mínimo:

- Componente para el filtro de nombre.
- Componente para el listado.
- Componente para la tarjeta de cada personaje del listado.
- Componente para el detalle de cada personaje.

Como en el ejemplo:



5. Detalle de personajes

Vamos a implementar una nueva funcionalidad: al hacer clic sobre la tarjeta de un personaje, su información aparecerá a pantalla completa. Para hacer esto usaremos rutas y la biblioteca **React Router**. En la pantalla de detalle aparecerá además de la foto, nombre, la casa a la que pertenece , si está vivo o muerto, género, especie y los nombres alternativos en caso de que los tenga

> **Nota:** no recomendamos mostrar el detalle del personaje con una ventana modal por encima del listado del personaje porque es más complejo. Es mejor mostrar el listado y los filtros o mostrar el componente `CharacterDetail` usando React Router.

6. Detallitos de calidad

- Como nos gusta cuidar la semántica, el campo de texto debe estar recubierto por una etiqueta `<form />`.
- Si estando en el campo de filtrado pulsamos intro debéis impedir que el navegador navegue o cambie la ruta sin querer.
- Si se busca por un texto por ejemplo "XXX" y no hay ningún personaje que coincida con dicho texto se debe mostrar un mensaje del tipo "No hay ningún personaje que coincida con la palabra XXX".
- El filtro debe filtrar independientemente de que la usuaria introduzca el texto en mayúsculas o minúsculas.
- Al entrar en el detalle de un personaje y a continuación pulsar atrás, el campo de texto debe mostrar el texto que tenía anteriormente.

7. BONUS: Mejoras visuales

Para terminar, podéis realizar algunas mejoras visuales del ejercicio. Por ejemplo:

- En el detalle del personaje mostrar la casa con su respectivo emblema y si un personaje está vivo o muerto con su respectivo icono.
- Usar algún sistema de grid para pintar el listado de personajes.
- Que funcione bien el responsive en dispositivos pequeños.

8. BONUS: URL compartible

- Como ejercicio extra, os proponemos que en el caso de que la usuaria navegue a una URL inexistente, como por ejemplo `http://localhost:3000/#/detail/12345` (el id 12345 no existe), debemos mostrar un mensaje del tipo "El personaje que buscas no existe".
- Y también debemos comtemplar que la URL del detalle de personaje sea compatible, es decir, que si visitamos esa URL directamente en el navegador se vea el detalle del personaje. Si refescamos el navegador en el detalle de un personaje (o cerrando y abriendo en la misma dirección) también debe volver a mostrar el detalle de ese personaje.

9. BONUS: Ordenación

Un extra interesante sería que ordenáseis el listado de personajes alfabéticamente por nombre.

10. BONUS: Más filtros

Un extra interesante sería que añadáis más filtros para filtrar por ejemplo por género.

11. BONUS: Boton Reset

Un extra interesante sería añadir un boton de reset para que la página vuelva a su listado principal

Entrega

Hemos pautado 9 horas de dedicación al ejercicio, el límite de entrega es

> jueves 04/12/2025 a las 23:59 horas

Solo debéis hacer commits y merges en la rama master de vuestro repositorio hasta la fecha límite. Si después del ejercicio queréis seguir trabajando sobre el ejercicio, lo podéis hacer en otra rama y no debéis mergearla hasta después de la entrevista técnica.

La evaluación solo se considerará terminada cuando:

- Esté publicada en GitHub Pages y esté funcionando, para lo cual tendréis que subir el código, también a la carpeta `docs/` del repositorio.
- El enlace a GitHub Pages esté en la página página principal del repositorio, en la parte superior, al lado de la descripción.

Normas

Este ejercicio está pensado para que lo realices de forma individual en clase, pero podrás consultar tus dudas con la profesora y tus compañeras si lo consideras necesario. Ellas no te darán directamente la solución de tu duda, pero sí pistas para poder solucionarla. Aún facilitando la comunicación entre compañeras, durante la prueba no debes copiar código de otra persona ni acceder a su portátil. Confiamos en tu responsabilidad.

La evaluación es una buena oportunidad para conocer cómo estás progresando, saber qué temas debes reforzar durante las siguientes semanas y cuáles dominas. Te recomendamos que te sientas cómoda con el ejercicio que entregues y no envíes cosas copiadas que no entiendas.

Si detectamos que has entregado código que no es tuyo, no entiendes y no lo puedes defender, pasarás directamente a la re-evaluación del módulo. Tu objetivo no debería ser pasar la evaluación sino convertirte en programadora, y esto debes tenerlo claro en todo momento.

Al final, tendrás un feedback sobre aspectos a destacar y a mejorar en tu ejercicio y sabrás qué objetivos de aprendizaje has superado de los listados a continuación.

Criterios de evaluación

Vamos a listar los criterios de evaluación de este ejercicio. Si no superas al menos el 80% de estos criterios o no has superado algún criterio clave (marcados con *) te pediremos que realices una re-evaluación con el fin de que termines el curso mejor preparada y enfrentes tu primera experiencia profesional con más seguridad. En caso contrario, estás aprendiendo al ritmo que hemos pautado para poder afrontar los conocimientos del siguiente módulo.

React básico

- Crea componentes con sintaxis correcta*.
- Crea una estructura adecuada de componentes*.
- Usa las props para pasar datos a componentes hijos*.
- Sabe pintar listados*.
- Sabe usar métodos funcionales de array (`map`, `filter`, etc.)*.
- Usa el estado para gestionar información de la interfaz*.
- El componente principal `App.js` maneja el estado de la aplicación*.
- Usa eventos en React para atender a interacciones del usuaria*.
- Hooks para las peticiones al servidor.
- Escribe un código sólido, sin errores en la consola*.
- Usa destructuring y valores por defecto en las props de los componentes.
- Usa `propTypes` para evitar errores de tipado.
- Tiene soltura a la hora de realizar cambios en el ejercicio presencial.

React router

- Crea rutas navegables dentro de una aplicación.

Otros criterios a tener en cuenta

- Usar inglés para nombres de variables, funciones, clases, comentarios, mensajes de commit, nombres de ficheros.
- El repositorio de GitHub debe tener README.md.

¡Expecto Patronum!!