

Game-Hub App Updated with React Query

App.tsx

```
import { Box, Flex, Grid, GridItem, Show } from "@chakra-ui/react";
import { useState } from "react";
import GameGrid from "./components/GameGrid";
import GameHeading from "./components/GameHeading";
import GenreList from "./components/GenreList";
import NavBar from "./components/NavBar";
import PlatformSelector from "./components/PlatformSelector";
import SortSelector from "./components/SortSelector";

// undefined: the absence of a value
// null: the intentional absence of a value

export interface GameQuery {
  genreId?: number;
  platformId?: number;
  sortOrder: string;
  searchText: string;
  pageSize: number;
}

function App() {
  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (
    <div>
      <Grid
        templateAreas={{
          // mobile devices
          base: `"nav" "main"`,

          // large devices, > 1024px
          lg: `"nav nav" "aside main"`,
        }}
        templateColumns={{
          base: "1fr",
          lg: "200px 1fr",
        }}
      >
        <GridItem area="nav">
```

```

<NavBar
  onSearch={(searchText) =>
    setGameQuery({ ...gameQuery, searchText })
  }
/>
</GridItem>
{/* will only be shown on lg screens and bigger */}
<Show above="lg">
  {" "}
<GridItem area="aside" paddingX={5}>
  <GenreList
    selectedGenreId={gameQuery.genreId}
    onSelectGenre={(genre) =>
      setGameQuery({ ...gameQuery, genreId: genre.id })
    }
  />
</GridItem>
</Show>
<GridItem area="main">
  <Box marginBottom={4}>
    <GameHeading gameQuery={gameQuery} />
    <Flex marginBottom={1}>
      <Box marginRight={5}>
        <PlatformSelector
          selectedPlatformId={gameQuery.platformId}
          onSelectPlatform={(platform) =>
            setGameQuery({ ...gameQuery, platformId: platform.id })
          }
        />
      </Box>
      <SortSelector
        sortOrder={gameQuery.sortOrder}
        onSelectSortOrder={(sortOrder) =>
          setGameQuery({ ...gameQuery, sortOrder })
        }
      />
    </Flex>
  </Box>
  <GameGrid gameQuery={gameQuery} />
</GridItem>
</Grid>
</div>
);
}

export default App;

```

useGames.ts

```
import { GameQuery } from "../App";
import { useInfiniteQuery } from "@tanstack/react-query";
import APIClient, { GetResponse } from "../services/api-client";
import { Platform } from "./usePlatforms";
import ms from "ms";

const apiClient = new APIClient<Game>('/games');

export interface Game {
  id: number;
  name: string;
  background_image: string;
  parent_platforms: { platform: Platform }[];
  metacritic: number;
  rating_top: number;
}

const useGames = (gameQuery: GameQuery) => useInfiniteQuery<GetResponse<Game>, Error>({
  queryKey: ['games', gameQuery],
  queryFn: ({ pageParam = 1 }) => apiClient.getAll({
    params: {
      genres: gameQuery.genreId,
      parent_platforms: gameQuery.platformId,
      ordering: gameQuery.sortOrder,
      search: gameQuery.searchText,
      page: pageParam,
    }
  }),
  getNextPageParam: (lastPage, allPages) => {
    return lastPage.next ? allPages.length + 1 : undefined;
  },
  staleTime: ms('1 day'),
}) ;

export default useGames
```

useGenres.ts

```
import { useQuery } from "@tanstack/react-query";
import ms from "ms";
import genres from "../data/genres";
import { APIClient } from "../services/api-client";

const apiClient = new APIClient<Genre>('/genres');

export interface Genre {
  id: number;
  name: string;
  image_background: string;
}

const useGenres = () => useQuery({
  queryKey: ['genres'],
  queryFn: apiClient.getAll,
  staleTime: ms('1 day'),
  initialData: genres,
});

export default useGenres;
```

useGenre.tsx

```
import useGenres from "./useGenres";

const useGenre = (id?: number) => {
  const { data: genres } = useGenres();
  return genres?.results.find((genre) => genre.id === id);
}

export default useGenre;
```

usePlatforms.ts

```
import { useQuery } from "@tanstack/react-query";
import APIClient, { GetResponse } from "../services/api-client";
import ms from "ms";
import platforms from "../data/platforms"

const apiClient = new APIClient<Platform>('/platforms/lists/parents');

export interface Platform {
    id: number;
    name: string;
    slug: string;
}

const usePlatforms = () => useQuery({
    queryKey: ['platforms'],
    queryFn: apiClient.getAll,
    staleTime: ms('1 day'),
    initialData: platforms,
}) 

export default usePlatforms;
```

usePlatform.ts

```
import usePlatforms from "./usePlatforms";

const usePlatform = (id?: number) => {
    const { data: platforms } = usePlatforms();
    return platforms?.results.find(
        (p) => p.id === id
    );
}

export default usePlatform;
```

api-client.ts

```
import axios, { AxiosRequestConfig } from "axios"

const axiosInstance = axios.create({
  baseURL: "https://api.rawg.io/api",
  params: {
    key: "c8ec0a603e934670b2bbbbd3f6d141c8"
  },
})

export interface GetResponse<T> {
  count: number;
  next: string | null;
  results: T[];
}

export class APIClient<T> {
  endpoint: string;

  constructor(endpoint: string) {
    this.endpoint = endpoint;
  }

  getAll = (config: AxiosRequestConfig) => {
    return axiosInstance
      .get<GetResponse<T>>(this.endpoint, config)
      .then(response => response.data)
  }
}

export default APIClient;
```

image-url.ts

```
import noImage from "../assets/no-image-placeholder-6f3882e0.webp"

const getCroppedImageUrl = (url: string) => {
  if (!url) return noImage;
  const target = 'media/';
  const index = url.indexOf(target) + target.length;
  return url.slice(0, index) + 'crop/600/400/' + url.slice(index);
}

export default getCroppedImageUrl;
```

GameGrid.tsx

```
import { SimpleGrid, Spinner, Text } from "@chakra-ui/react";
import GameCard from "./GameCard";
import GameCardSkeleton from "./GameCardSkeleton";
import GameCardContainer from "./GameCardContainer";
import { GameQuery } from "../App";
import useGames from "../hooks/useGames";
import React from "react";
import InfiniteScroll from "react-infinite-scroll-component";

interface Props {
  gameQuery: GameQuery;
}

const GameGrid = ({ gameQuery }: Props) => {
  const {
    data,
    error,
    isLoading,
    isFetchingNextPage,
    fetchNextPage,
    hasNextPage,
  } = useGames(gameQuery);
  const skeletons = [1, 2, 3, 4, 5, 6];

  if (error) return <Text>{error.message}</Text>

  const fetchedGamesCount =
    data?.pages.reduce((total, page) => total + page.results.length, 0) || 0;

  return (
    // wrap the SimpleGrid in InfiniteScroll
    <InfiniteScroll
      // uses the accumulated number of games fetched so far
      dataLength={fetchedGamesCount}
      next={() => fetchNextPage()}
      // expects a boolean value, so if hasNextPage is undefined, it will be false
      hasMore={!hasNextPage}
      loader={<Spinner />}
      endMessage={
        <p style={{ textAlign: "center" }}>
          <b>Yay! You have seen it all</b>
        </p>
      }
    >
      {data?.results.map((game) => (
        <GameCard key={game.id} game={game} />
      ))}
      {skeletons.map(() => (
        <GameCardSkeleton />
      ))}
    </InfiniteScroll>
  );
}
```

```

        }
      >
      <SimpleGrid columns={{ sm: 1, md: 2, lg: 3, xl: 4 }} spacing={6}>
        {isLoading &&
          skeletons.map((skeleton) =>
            <GameCardContainer key={skeleton}>
              <GameCardSkeleton />
            </GameCardContainer>
          ) )
        }

        /* map each page to a React Fragment,
         then map each game to a GameCardContainer */
      {data?.pages.map((page, index) =>
        <React.Fragment key={index}>
          {page.results.map((game) =>
            <GameCardContainer key={game.id}>
              <GameCard game={game} />
            </GameCardContainer>
          ) )
        </React.Fragment>
      ) )
    </SimpleGrid>
  </InfiniteScroll>
);
};

export default GameGrid;

```

GameHeading.tsx

```

import { Heading } from "@chakra-ui/react";
import { GameQuery } from "../App";
import useGenres from "../hooks/useGenres";
import usePlatform from "../hooks/usePlatform";
import useGenre from "../hooks/useGenre";

interface Props {
  gameQuery: GameQuery;
}

const GameHeading = ({ gameQuery }: Props) => {
  const genre = useGenre(gameQuery.genreId);

```

```

const { data: genres } = useGenres();
const platform = usePlatform(gameQuery.platformId);

const heading = `${platform?.name || ""} ${genre?.name || ""} Games`;
return (
  <Heading as="h1" marginY={5} fontSize="5xl">
    {heading}
  </Heading>
);
};

export default GameHeading;

```

PlatformSelector.tsx

```

import { Button, Menu, MenuButton, MenuItem, MenuList } from "@chakra-ui/react";
import { BsChevronDown } from "react-icons/bs";
import usePlatforms, { Platform } from "../hooks/usePlatforms";
import usePlatform from "../hooks/usePlatform";

interface Props {
  onSelectPlatform: (platform: Platform) => void;
  selectedPlatformId?: number;
}

const PlatformSelector = ({ onSelectPlatform, selectedPlatformId }: Props) => {
  const { data, error } = usePlatforms();
  const selectedPlatform = usePlatform(selectedPlatformId);

  if (error) return null;
  return (
    <Menu>
      <MenuButton as={Button} rightIcon={<BsChevronDown />}>
        {selectedPlatform?.name || "Platforms"}
      </MenuButton>
      <MenuList>
        {data?.results.map((platform) => (
          <MenuItem
            onClick={() => onSelectPlatform(platform)}
            key={platform.id}
          >
            {platform.name}
          </MenuItem>
        ))}
      </MenuList>
    </Menu>
  );
}

```

