GAMEHUB 05-09-23

index.html

```html
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <link rel="icon" type="image/svg+xml" href="" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Evan Marie Carr: GameHub</title>
 </head>
 <body>
   <div id="root"></div>
   <script type="module" src="/src/main.tsx"></script>
 </body>
</html>
```

index.css

```css
html,
body {
 font-family: monospace !important;
}

form {
 width: 100%;
}
```

theme.ts

```
import { extendTheme, ThemeConfig } from "@chakra-ui/react"

/* info on Chakra Color Mode:
https://chakra-ui.com/docs/styled-system/color-mode */

const config: ThemeConfig = {
   initialColorMode: "dark"
};

const theme = extendTheme({ config,
   // creating custom color palette with tools listed on Chakra site
colors: {
   gray: {
       50: '#f9f9f9',
       100: '#ededed',
       200: '#d3d3d3',
       300: '#b3b3b3',
       400: '#a0a0a0',
       500: '#898989',
       600: '#6c6c6c',
       700: '#202020',
       800: '#121212',
       900: '#111',

   }

} });

export default theme;
```

This code defines a custom Chakra UI theme by extending the base Chakra theme with custom settings. The extendTheme() function is imported from the @chakra-ui/react library, and it takes an object with the custom theme configuration as an argument.

The custom configuration includes an object called config, which specifies the initial color mode for the theme. In this case, it sets the initial color mode to "dark". The custom configuration also includes an object called colors, which defines a custom color palette for the theme. In this case, it defines a "gray" color palette with ten shades of gray, from light to dark.

Finally, the export default statement at the end of the code exports the custom theme object so that it can be used in other parts of the application.

main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import { ChakraProvider, ColorModeScript } from "@chakra-ui/react";
import App from "./App";
import theme from "./theme";
import "./index.css";

// RAWG API-Key = c8ec0a603e934670b2bbbbd3f6d141c8

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(
 <React.StrictMode>
   <ChakraProvider theme={theme}>
     <ColorModeScript initialColorMode={theme.config.initialColorMode} />
     <App />
   </ChakraProvider>
 </React.StrictMode>
);
```

App refers to the main component of the React application. theme refers to the custom Chakra UI theme that will be applied to the application.

<React.StrictMode>: A component that activates additional strict checks and warnings for the React application.

<ChakraProvider>: A component that applies the custom Chakra UI theme to the React application.

<ColorModeScript>: A component that sets the initial color mode for the custom Chakra UI theme.

App.css

```
#root {
 max-width: 1280px;
 margin: 0 auto;
 padding: 2rem;
```

The code selects the #root element using the CSS ID selector, and sets the maximum width to 1280 pixels, centers it on the page, and adds 2em of padding. The text-align property centers the text within the #root element.

```
 text-align: center;
}

.logo {
 height: 6em;
 padding: 1.5em;
 will-change: filter;
 transition: filter 300ms;
}
.logo:hover {
 filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
 filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
 from {
   transform: rotate(0deg);
 }
 to {
   transform: rotate(360deg);
 }
}

@media (prefers-reduced-motion: no-preference) {
 a:nth-of-type(2) .logo {
   animation: logo-spin infinite 20s linear;
 }
}

.card {
 padding: 2em;
}
```

The code selects the `.logo` class, which presumably applies to an image element. The `height` property sets the height of the image to 6em, and the `padding` property adds 1.5em of padding around the image. The `will-change` property tells the browser to optimize for changes in the `filter` property, which is set by the `:hover` pseudo-class. The `transition` property specifies the duration and timing function for the `filter` transition.

The code adds a `:hover` pseudo-class to the `.logo` class. When the image is hovered over, the `filter` property is set to `drop-shadow(0 0 2em #646cffaa)`. This creates a blue drop shadow effect around the image.

The code adds a `:hover` pseudo-class to the `.logo.react` class. When the image is hovered over and has the `.react` class, the `filter` property is set to `drop-shadow(0 0 2em #61dafbaa)`. This creates a green drop shadow effect around the image.

The code defines a `@keyframes` animation called `logo-spin`. This animation rotates an

```
.read-the-docs {
 color: #888;
}
```

App.tsx

```
import React, { useState } from "react";
import {
 Box,
 Button,
 ButtonGroup,
 Flex,
```

element from 0 degrees to 360 degrees.

The code uses a media query to check whether the user has expressed a preference for reduced motion using the `prefers-reduced-motion` media feature. If the user has no preference for reduced motion, the animation is applied to the second link in a set of links, which presumably has a `.logo` class. The animation `logo-spin` is applied with a duration of 20 seconds and a linear timing function.

The code selects the `.card` class and adds 2em of padding.

The code selects an element with the class `read-the-docs` and sets the text color to a muted gray (#888).

```
  Grid,
  GridItem,
  HStack,
  Show,
} from "@chakra-ui/react";
import NavBar from "./components/NavBar";
import GameGrid from "./components/GameGrid";
import GenreList from "./components/GenreList";
import { Genre } from "./hooks/useGenres";
import PlatformSelector from "./components/PlatformSelector";
import { Platform } from "./hooks/useGames";
import SortSelector from "./components/SortSelector";
import GameHeading from "./components/GameHeading";

export interface GameQuery {
  genre: Genre | null;
  platform: Platform | null;
  sortOrder: string;
  searchText: string;
}

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

  return (
    <div>

      {/* Grid areas before setting to be responsive:
      <Grid templateAreas={`"nav  nav" "aside  main"`}>
      MORE INFO ON RESPONSIVE STYLES:
      ** https://chakra-ui.com/docs/styled-system/responsive-styles ** */}

      <Grid
        templateAreas={{
```

1. The code defines an interface called GameQuery, which describes the properties of a game search query. The GameQuery interface has four properties: genre, platform, sortOrder, and searchText.
2. The App component is defined as a function using the useState hook to manage state. The initial state is an empty object that conforms to the GameQuery interface.
3. The return statement of the App function defines the layout and components of the game search interface. The interface consists of a Grid with three GridItems: nav, aside, and main.
4. The nav area contains a NavBar component, which accepts a callback function as a prop to handle search queries. When a search query is entered, the setGameQuery function is called to update the game query state.
5. The aside area is only displayed on screens larger than 1024px. It contains

```
      // 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
          selectedGenre={gameQuery.genre}
          onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}
        />
      </GridItem>
    </Show>
    <GridItem area="main">
      <Box paddingLeft={9}>
        <GameHeading gameQuery={gameQuery} />
        <Flex marginBottom={1}>
          <Box marginRight={5}>
            <PlatformSelector
```

a GenreList component, which accepts a callback function as a prop to handle genre selections. When a genre is selected, the setGameQuery function is called to update the game query state.

6. The main area contains a GameHeading component, a PlatformSelector component, a SortSelector component, and a GameGrid component.
7. The GameHeading component displays the current search query parameters.
8. The PlatformSelector component allows the user to select a platform for their search, and calls setGameQuery to update the game query state.
9. The SortSelector component allows the user to select a sorting order for their search, and calls setGameQuery to update the game query state.
10. The GameGrid component displays a grid of game results based on the current search query.

Genre Selection Explained:

1) GenreList has the command onSelectGenre

```
                  selectedPlatform={gameQuery.platform}
                  onSelectPlatform={(platform) =>
                    setGameQuery({ ...gameQuery, platform })
                  }
                />
              </Box>
              <SortSelector
                sortOrder={gameQuery.sortOrder}
                onSelectSortOrder={(sortOrder) =>
                  setGameQuery({ ...gameQuery, sortOrder })
                }
              />
            </Flex>
          </Box>
          <GameGrid gameQuery={gameQuery} />
        </GridItem>
      </Grid>
    </div>
  );
}

export default App;
```

ColorModeSwitch.tsx

```
import { HStack, Switch, Text, useColorMode } from "@chakra-ui/react";

const ColorModeSwitch = () => {
 const { toggleColorMode, colorMode } = useColorMode();
 return (
   <HStack>
     <Switch
       colorScheme="green"
```

which notifies the parent, App.tsx, that a genre has been selected.

2) App.tsx gets notified a genre has been selected and setSelectedGenre passes the genre that has been selected.

3) The causes the App component to re-render, at which point, the newly selected genre is passed to the GameGrid in the next render

4) The GameGrid takes the selected genre and passes it to the useGames hook

5) useGames passes the selected genre to the useData hook as a query parameter

6) The useData hook is flexible in that it takes parameters and data requests and an array of dependencies, any changes in which, causes the effect to rerun and refetch the data from the server

1. The ColorModeSwitch component is defined as an arrow function that takes no arguments.
2. The useColorMode hook is called to access the current color mode and the toggleColorMode function, which toggles the color mode between light and dark.
3. The return statement of the

```
        isChecked={colorMode === "dark"}
        onChange={toggleColorMode}
      />
      <Text whiteSpace="nowrap">color mode</Text>
    </HStack>
  );
};

export default ColorModeSwitch;
```

ColorModeSwitch component returns an HStack component, which contains a Switch component and a Text component.

4. The Switch component allows the user to toggle between light and dark color modes. It is styled with the green color scheme and is checked if the current color mode is "dark". When the user toggles the switch, the toggleColorMode function is called to update the color mode.
5. The Text component displays the label "color mode" next to the Switch component.

Emoji.tsx

```
import bullsEye from "../assets/bulls-eye.webp";
import thumbsUp from "../assets/thumbs-up.webp";
import meh from "../assets/meh.webp";
import { Image, ImageProps } from "@chakra-ui/react";

interface Props {
 rating: number;
}

const Emoji = ({ rating }: Props) => {
```

1. The code imports three image files as WebP files using relative paths. It also imports the Image and ImageProps components from the @chakra-ui/react library.
2. The Emoji component is defined as an arrow function that takes a single prop, rating, of type number.
3. The if statement checks if the rating prop is less than 3. If it is, the

```
  if (rating < 3) return null;

  // index signature: object can have any number of keys, which are numbers
  const emojiMap: { [key: number]: ImageProps } = {
    3: { src: meh, alt: "meh", boxSize: "20px" },
    4: { src: thumbsUp, alt: "recommended", boxSize: "20px" },
    5: { src: bullsEye, alt: "exceptional", boxSize: "25px" },
  };
  return <Image {...emojiMap[rating]} marginTop={1} />;
};

export default Emoji;
```

function returns `null` and no emoji is displayed.

4. The `emojiMap` object is defined using an index signature, which allows the object to have any number of keys that are of the `number` type. The keys correspond to possible rating values (3, 4, and 5), and the values are objects with properties for the image source, alt text, and box size.
5. The `return` statement of the `Emoji` component returns an `Image` component with props that are determined by the `rating` prop. The `...emojiMap[rating]` syntax is used to spread the properties of the corresponding object from the `emojiMap` object into the `Image` component. The `marginTop` property is also added to the `Image` component to create some vertical space below the emoji.

GameCard.tsx

```
import React from "react";
import { Game } from "../hooks/useGames";
import { Card, CardBody, HStack, Heading, Image, Text } from
"@chakra-ui/react";
import PlatformIcons from "./PlatformIcons";
import CriticScore from "../assets/CriticScore";
import getCroppedImageUrl from "../services/image-url";
import Emoji from "./Emoji";

interface Props {
 game: Game;
}

const GameCard = ({ game }: Props) => {
 return (
   <Card>
     <Image src={getCroppedImageUrl(game.background_image)} />
     <CardBody>
       <HStack justifyContent="space-between" marginBottom={3}>
         <PlatformIcons
           platforms={game.parent_platforms?.map((p) => p.platform)}
         />
         <CriticScore score={game.metacritic} />
       </HStack>
       <Heading fontSize="2xl" fontFamily="monospace">
         {game.name}
         <Emoji rating={game.rating_top} />
       </Heading>
     </CardBody>
   </Card>
```

1. The code defines an interface called Props, which describes the properties of the GameCard component. The Props interface has a single property: game, which is of type Game.
2. The GameCard component is defined as an arrow function that takes a single prop, game, of type Game.
3. The return statement of the GameCard function returns a Card component, which contains an Image component and a CardBody component.
4. The Image component displays a cropped image of the game's background image, which is obtained by calling a function called getCroppedImageUrl with the background_image property of the game object.
5. The CardBody component contains a HStack component and a Heading component.
6. The HStack component contains a PlatformIcons component and a CriticScore component. The PlatformIcons component displays icons

```
 );
};

export default GameCard;
```

for the game's parent platforms, which are obtained from the parent_platforms property of the game object. The CriticScore component displays the game's Metacritic score, which is obtained from the metacritic property of the game object.

7. The Heading component displays the name of the game, obtained from the name property of the game object. The Emoji component is also included in the heading, which displays an emoji based on the game's rating_top property.

GameCardContainer.tsx

```
import { Box } from "@chakra-ui/react";
import { ReactNode } from "react";

interface Props {
 children: ReactNode;
}

const GameCardContainer = ({ children }: Props) => {
 return (
   <Box borderRadius={10} overflow="hidden">
     {children}
```

1. The code imports the Box component from the @chakra-ui/react library and the ReactNode type from the react library.
2. The Props interface is defined with a single property: children, which is of type ReactNode. ReactNode is a type that includes all valid children types in a React component, such as JSX elements or null.
3. The GameCardContainer component is defined as an arrow function that takes

```
    </Box>
  );
};

export default GameCardContainer;
```

a single prop, children, of type ReactNode.

4. The return statement of the GameCardContainer function returns a Box component with properties for border radius and overflow. The Box component contains the children prop, which will be the GameCard component.

GameCardSkeleton.tsx

```
import { Card, CardBody, Skeleton, SkeletonText } from "@chakra-ui/react";

const GameCardSkeleton = () => {
  return (
    <Card>
      <Skeleton height="200px" />
      <CardBody>
        <SkeletonText />
      </CardBody>
    </Card>
  );
};

export default GameCardSkeleton;
```

1. The GameCardSkeleton component is defined as an arrow function that takes no arguments.
2. The return statement of the GameCardSkeleton function returns a Card component, which contains a Skeleton component and a CardBody component.
3. The Skeleton component displays a placeholder image with a height of 200px.
4. The CardBody component contains a SkeletonText component, which displays a placeholder text block.
5. The GameCardSkeleton component is exported as the default export of the

module, so it can be imported and used in other parts of the application.

GameGrid.tsx

```tsx
import { SimpleGrid, Text } from "@chakra-ui/react";
import useGames, { Platform } from "../hooks/useGames";
import GameCard from "./GameCard";
import GameCardSkeleton from "./GameCardSkeleton";
import GameCardContainer from "./GameCardContainer";
import { Genre } from "../hooks/useGenres";
import { GameQuery } from "../App";

interface Props {
 gameQuery: GameQuery;
}

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

 if (error) return <Text>{error}</Text>;

 return (
   <SimpleGrid
     columns={{ sm: 1, md: 2, lg: 3, xl: 4 }}
     padding={10}
     spacing={6}
   >
     {isLoading &&
       skeletons.map((skeleton) => (
         <GameCardContainer key={skeleton}>
```

1. The code imports the useGames, SimpleGrid, Text, GameCard, GameCardContainer, and GameCardSkeleton components, as well as the Props interface, from other parts of the application.
2. The GameGrid component is defined as an arrow function that takes a single prop, gameQuery, of type GameQuery.
3. The useGames hook is called with the gameQuery prop to fetch data about the games that match the query. The data, error, and isLoading variables are destructured from the result of the hook.
4. An array of skeletons is defined with 6 elements, which will be used to display placeholder GameCard components while the data is being fetched.
5. An if statement checks if there is an error while fetching the data. If there

```
            <GameCardSkeleton />
          </GameCardContainer>
        ))}
      {data.map((data) => (
        <GameCardContainer key={data.id}>
          <GameCard game={data} />
        </GameCardContainer>
      ))}
    </SimpleGrid>
 );
};

export default GameGrid;
```

   is, a Text component is returned with the error message.
6. The return statement of the GameGrid function returns a SimpleGrid component, which displays a grid of GameCardContainer components.
7. If the data is still loading, the skeletons array is mapped over to display GameCardContainer components with GameCardSkeleton components inside them.
8. If the data has been loaded, the data array is mapped over to display GameCardContainer components with GameCard components inside them, passing the game object as a prop to each GameCard.

GameHeading.tsx

```
import { Heading } from "@chakra-ui/react";
import { GameQuery } from "../App";

interface Props {
 gameQuery: GameQuery;
}
```

1. The GameHeading component is defined as an arrow function that takes a single prop, gameQuery, of type GameQuery.
2. The heading variable is defined using template literals to concatenate the names of the selected platform and genre from the gameQuery prop. If

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

export default GameHeading;
```

GenreList.tsx

```
import {
 Button,
 HStack,
 Heading,
 Image,
 List,
 ListItem,
 Spinner,
 Text,
} from "@chakra-ui/react";
import useGenres, { Genre } from "../hooks/useGenres";
import getCroppedImageUrl from "../services/image-url";

interface Props {
```

either of these values are null, an empty string is used instead.

3. The return statement of the GameHeading function returns a Heading component, which displays the heading text as an h1 element. The marginY and fontSize props are set to provide appropriate styling.
4. The GameHeading component is exported as the default export of the module, so it can be imported and used in other parts of the application.

1. The code imports the useGenres, Image, List, ListItem, Button, Heading, HStack, Spinner, Genre, and Props from other parts of the application.
2. The GenreList component is defined as an arrow function that takes two props, selectedGenre and onSelectGenre, both of type Genre or null.
3. The useGenres hook is called to fetch a list of genres from the API. The data, isLoading, and error variables are destructured from the result of the hook.

```
 onSelectGenre: (genre: Genre) => void;
 selectedGenre: Genre | null;
}

const GenreList = ({ selectedGenre, onSelectGenre }: Props) => {
 const { data, isLoading, error } = useGenres();

 if (error) return null;
 // not using spinner, but keeping in case retrieval of genres changes
 if (isLoading) return <Spinner />;

 return (
   <>
     <Heading fontSize="2xl" marginBottom={3}>
       Genres
     </Heading>
     <List>
       {data.map((data) => (
         <ListItem
           onClick={() => onSelectGenre(data)}
           key={data.id}
           paddingY="5px"
           paddingX="5px"
           borderRadius="5px"
           _hover={{
             backgroundColor: "purple.500",
             color: "cyan",
             fontWeight: "bold",
           }}
         >
           <HStack>
             <Image
               boxSize="32px"
               borderRadius={8}
```

4. If there is an error while fetching the data, `null` is returned.
5. If the data is still loading, a `Spinner` component is displayed while the data is being fetched.
6. If the data has been loaded, the `return` statement of the `GenreList` function returns a `List` component that displays a list of genres.
7. The `data` array is mapped over to display a `ListItem` component for each genre. Each `ListItem` component is wrapped in an `HStack` component, which displays the genre image and name.
8. A `Button` component is used to display the genre name. The `onClick` prop is set to call `onSelectGenre` with the selected genre when the button is clicked.
9. If the genre is currently selected, the `fontWeight`, `color`, and `textDecoration` props are set to highlight the selected genre.
10. If the genre name is too long, the `whiteSpace` and `textAlign` props are set to wrap the text and align it to the left.

```
            // image will fill the container while preserving aspect ratio
            objectFit="cover"
            src={getCroppedImageUrl(data.image_background)}
            onClick={() => onSelectGenre(data)}
          />{" "}
          <Button
            // to fix extra long genre names
            whiteSpace="normal"
            textAlign="left"
            fontWeight={data.id === selectedGenre?.id ? "bold" : "normal"}
            onClick={() => onSelectGenre(data)}
            fontSize="xl"
            color={data.id === selectedGenre?.id ? "cyan" : ""}
            textDecoration={
              data.id === selectedGenre?.id ? "underline" : ""
            }
            variant="link"
            _hover={{
              textDecoration: "",
              fontWeight: "bold",
            }}
          >
            {data.name === "Massively Multiplayer" ? "MMO" : data.name}
          </Button>
        </HStack>
      </ListItem>
    ))}
  </List>
 </>
 );
};

export default GenreList;
```

11. The return statement of the GenreList function returns the List component wrapped in a Heading component that displays the text "Genres" and the list of genres.
12. The GenreList component is exported as the default export of the module, so it can be imported and used in other parts of the application.

NavBar.tsx

```tsx
import { HStack, Image } from "@chakra-ui/react";
import logo from "../assets/logo.webp";
import ColorModeSwitch from "./ColorModeSwitch";
import SearchInput from "./SearchInput";

interface Props {
 onSearch: (searchText: string) => void;
}

const NavBar = ({ onSearch }: Props) => {
 return (
   <HStack padding="10px" marginRight={3}>
     <Image src={logo} boxSize="60px" />
     <SearchInput onSearch={onSearch} />
     <ColorModeSwitch />
   </HStack>
 );
};

export default NavBar;
```

1. The code imports the Image, HStack, logo, SearchInput, ColorModeSwitch, and Props from other parts of the application.
2. The NavBar component is defined as an arrow function that takes one prop, onSearch, a function that takes a string argument.
3. The return statement of the NavBar function returns an HStack component that displays the logo, a search input, and the color mode switch.
4. The Image component is used to display the logo.
5. The SearchInput component is used to allow the user to search for games. The onSearch prop is set to call the onSearch function with the search text when the user submits the search.
6. The ColorModeSwitch component is used to allow the user to switch between light and dark mode.

PlatformIcons.tsx

```tsx
import {
 FaWindows,
 FaPlaystation,
 FaXbox,
 FaApple,
 FaLinux,
 FaAndroid,
} from "react-icons/fa";
import { MdPhoneIphone } from "react-icons/md";
import { SiPlaystation3, SiNintendo } from "react-icons/si";
import { BsGlobe, BsPlaystation } from "react-icons/bs";
import { Platform } from "../hooks/useGames";
import { HStack, Icon } from "@chakra-ui/react";
import { IconType } from "react-icons";

interface Props {
 platforms: Platform[];
}

const PlatformIcons = ({ platforms = [] }: Props) => {
 // index signature telling TypeScript how the iconMap is structured
 const iconMap: { [key: string]: IconType } = {
   // slug is a textual id, all lowercase
   pc: FaWindows,
   playstation: FaPlaystation,
   xbox: FaXbox,
   nintendo: SiNintendo,
   mac: FaApple,
   linux: FaLinux,
   android: FaAndroid,
```

1. The code imports the HStack, Icon, IconType, and Props from other parts of the application.
2. The PlatformIcons component is defined as an arrow function that takes one prop, platforms, an array of Platform objects.
3. The return statement of the PlatformIcons function returns an HStack component that displays the icons for each platform in the platforms prop.
4. An iconMap object is defined to map platform slugs to the corresponding icon. Each key is a string (the platform slug) and each value is an IconType (the corresponding icon).
5. A loop is used to iterate over each Platform object in the platforms prop. For each Platform, an Icon component is rendered with the corresponding icon from the iconMap. The Icon component is styled with the color prop set to

```
    ios: MdPhoneIphone,
    web: BsGlobe,
  };

  /* chakra style system (gray.500)
  https://v1.chakra-ui.com/docs/styled-system/theming/theme?scroll=true
  marginy = horizontal margin, using numerical values is best.
  */

  return (
    <HStack marginY={1}>
      {platforms.map((platform) => (
        <Icon key={platform.id} as={iconMap[platform.slug]} color="gray.500" />
      ))}
    </HStack>
  );
};

export default PlatformIcons;
```

"gray.500" to ensure that the icons have a consistent color.

6. The PlatformIcons component is exported as the default export of the module, so it can be imported and used in other parts of the application.

PlatformSelector.tsx

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

interface Props {
  onSelectPlatform: (platform: Platform) => void;
  selectedPlatform: Platform | null;
}
```

This is a React functional component called PlatformSelector. It is a dropdown menu component that allows users to select a gaming platform, such as PlayStation or Xbox. The component takes two props: onSelectPlatform, a function that will be called when a platform is selected, and selectedPlatform, the currently selected

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

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

export default PlatformSelector;
```

platform.

The component uses the usePlatforms hook to fetch a list of platforms from an API. If there is an error retrieving the platforms, the component will not render anything. Otherwise, the component will render a Menu from the Chakra UI library. The menu contains a MenuButton, which is a styled Button component that displays the currently selected platform name and a chevron-down icon. When clicked, the MenuButton will open a MenuList, which is a list of MenuItems, each representing a gaming platform.

When a MenuItem is clicked, the onSelectPlatform function is called with the corresponding Platform object as its argument, and the selectedPlatform state is updated to the selected platform.

SearchInput.tsx

```
import { Input, InputGroup, InputLeftElement } from "@chakra-ui/react";
import { useRef } from "react";
import { BsSearch } from "react-icons/bs";
```

This code defines a functional component named SearchInput that takes one prop named

```tsx
interface Props {
 onSearch: (searchText: string) => void;
}

const SearchInput = ({ onSearch }: Props) => {
 const ref = useRef<HTMLInputElement>(null);
 return (
   <form
     onSubmit={(event) => {
       event.preventDefault();
       if (ref.current) onSearch(ref.current.value);
     }}
   >
     <InputGroup>
       <InputLeftElement children={<BsSearch />} />
       <Input
         ref={ref}
         borderRadius={20}
         placeholder="Search games..."
         variant="filled"
       />
     </InputGroup>
   </form>
 );
};

export default SearchInput;
```

onSearch, which is a function that takes a string parameter. The component imports two Chakra-UI components: InputGroup and Input, and one icon from React-icons: BsSearch. The component uses the useRef hook to create a reference to the input element to access its value. It then returns a form element that handles the search functionality. When the form is submitted, the onSubmit event is triggered, preventing the default action (page refresh). The component then checks if the ref to the input element exists and if it does, it calls the onSearch function with the current value of the input element as an argument. The search input field is displayed with the search icon on the left and the placeholder text "Search games..." within the input field.

SortSelector.tsx

```tsx
import { Button, Menu, MenuButton, MenuItem, MenuList } from
"@chakra-ui/react";
```

This code exports a React functional

```
import { BsChevronDown } from "react-icons/bs";

interface Props {
 onSelectSortOrder: (sortOrder: string) => void;
 sortOrder: string;
}

const SortSelector = ({ onSelectSortOrder, sortOrder }: Props) => {
 // hyphen indicates reversed order, highest rating, newest games etc first
 const sortOrders = [
   { value: "", label: "relevance " },
   { value: "-added", label: "date added " },
   { value: "name", label: "name " },
   { value: "-released", label: "release date " },
   { value: "-metacriti", label: "popularity " },
   { value: "-rating", label: "average rating " },
 ];

 const currentSortOrder = sortOrders.find(
   (order) => order.value === sortOrder
 );
 return (
   <Menu>
     <MenuButton as={Button} rightIcon={<BsChevronDown />}>
       Sort by: {currentSortOrder?.label || "relevance"}
     </MenuButton>
     <MenuList>
       {sortOrders.map((order) => (
         <MenuItem
           onClick={() => onSelectSortOrder(order.value)}
           key={order.value}
           value={order.value}
         >
           {order.label}
```

component named SortSelector that takes two props: onSelectSortOrder, which is a function that accepts a string as an argument, and sortOrder, which is a string. The component creates a drop-down menu with a label "Sort by:" and a current sort order. The current sort order is determined by finding the first element in an array of sort order objects (sortOrders) that has a value matching the sortOrder prop. If no match is found, the default value is "relevance". The menu items are created by mapping through the sortOrders array and creating a MenuItem component for each object. When a menu item is clicked, the component calls the onSelectSortOrder function with the value of the clicked item as the argument.

```
          </MenuItem>
        ))}
      </MenuList>
    </Menu>
  );
};

export default SortSelector;
```

useData.ts

```
import { useEffect, useState } from "react";
import apiClient from "../services/api-client";
import { AxiosRequestConfig, CanceledError } from "axios";

interface GetResponse<T> {
   count: number;
   results: T[];
}

const useData = <T>(endpoint: string, requestConfig?: AxiosRequestConfig,
deps?: any[]) => {
 const [data, setData] = useState<T[]>([]);
 const [error, setError] = useState("");
 const [isLoading, setLoading] = useState(false)

 useEffect(() => {
```

This code exports a custom React hook named useData that takes two generic type parameters T and endpoint as well as two optional parameters requestConfig and deps.

The hook initializes three state variables: data, error, and isLoading, which are initially set to an empty array, an empty string, and false, respectively.

The hook then uses the useEffect hook to make a GET request to an API endpoint using axios. The hook sets the isLoading state to true

```
    const controller = new AbortController();
    setLoading(true);
    apiClient
      .get<GetResponse<T>>(endpoint, { signal: controller.signal,
...requestConfig })
      .then((response) => {
        setData(response.data.results);
        setLoading(false);
      })

      .catch((error) => {
        if (error instanceof CanceledError) return;
        setError(error.message)
        setLoading(false);
      })

    // cleanup function:
    return () => controller.abort();
  }, deps ? [...deps] : []);
  return { data, error, isLoading };
};

export default useData;
```

before making the API request and uses AbortController to cancel the request in case the component is unmounted before the request is completed. If the request is successful, the hook sets the data state to the results property of the response data and sets isLoading to false. If the request fails, the hook sets the error state to the error message and sets isLoading to false.

The hook returns an object with three properties: data, error, and isLoading, which can be used in a React component.

useGames.ts

```
import { GameQuery } from "../App";
import useData from "./useData";
import { Genre } from "./useGenres";

export interface Platform {
   id: number;
   name: string;
```

This code exports a custom hook named useGames which takes a GameQuery object as a parameter. The hook internally calls another custom hook useData to fetch a list of games from the "/games" endpoint using the Axios

```
    slug: string;
}

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

const useGames = (
  gameQuery: GameQuery) =>
  useData<Game>(
    "/games",
  {
  params: {
    genres: gameQuery.genre?.id,
    platforms: gameQuery.platform?.id,
    ordering: gameQuery.sortOrder,
    search: gameQuery.searchText
  }},
    [gameQuery]);

export default useGames
```

library. The useData hook uses the provided request configuration to make the GET request, and returns the response data, error, and loading state. The useGames hook adds query parameters to the GET request such as genre, platform, sort order, and search text based on the GameQuery object. Finally, the useGames hook returns the data, error, and loading state returned by useData hook after customizing the GET request based on the gameQuery parameter.

useGenres.ts

```
import genres from "../data/genres";

export interface Genre {
    id: number;
```

This code exports a custom hook named useGenres which returns an object with three properties: data, isLoading, and error. This

```
   name: string;
   image_background: string;
}

/* returning object with 3 properties to minimize the impact on components
which are consumers of this hook. Genres are available right away, no
spinner necessary */

const useGenres = () => ({ data: genres, isLoading: false, error: null} );

export default useGenres;
```

usePlatform.ts

```
import useData from "./useData";

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

const usePlatforms = () => useData<Platform>('/platforms/lists/parents');

export default usePlatforms;
```

hook does not use any external libraries like Axios or Chakra UI. Instead, it simply returns an object with dummy data for `data` and `isLoading` set to `false` and `error` set to `null`. This is done to minimize the impact on components that consume this hook. Since the genres are available right away, no spinner is necessary.

This code defines a custom React hook named `usePlatforms` which makes use of the `useData` hook to fetch a list of video game platforms from an API endpoint.

The hook imports the `useData` hook and exports the `usePlatforms` hook. The `usePlatforms` hook calls the `useData` hook with the endpoint URL as the argument and `Platform` type parameter as a type argument.

The `useData` hook will make a GET request to the specified endpoint using the `axios` library to fetch data from the API. It takes in three arguments, `endpoint`, `requestConfig`, and `deps`. In this case, `endpoint` is set to `'/platforms/lists/parents'` to fetch the list of video game platforms from the API.

Once the data is fetched, the useData hook returns an object containing three properties - data, error, and isLoading. The data property contains the list of video game platforms fetched from the API, the error property contains any error messages that occurred during the API request, and the isLoading property indicates whether the API request is still in progress.

Finally, the usePlatforms hook returns the result of calling the useData hook, which is an object with data, error, and isLoading properties.

app-client.ts

```
import axios from "axios"

export default axios.create({
   baseURL: "https://api.rawg.io/api",
   params: {
       key: "c8ec0a603e934670b2bbbbd3f6d141c8"
   },
})
```

This code exports an Axios instance that is preconfigured to make requests to the RAWG API. It sets the base URL for the API to "https://api.rawg.io/api" and sets a default parameter of an API key that is required for accessing the API. This API key is passed as a query parameter in all requests made using this instance.

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;
```

This code exports a utility function named getCroppedImageUrl that takes a string URL as input and returns a modified URL with "crop/600/400/" added after the "media/" part of the original URL. If the input URL is falsy, the function returns a default "no image" placeholder image URL. This function is used to crop and resize images fetched from the RAWG API.

genres.ts

```
// Static Data: all genres retrieved from server
export default
[
 {
   "id": 4,
   "name": "Action",
   "slug": "action",
   "games_count": 172395,
   "image_background":
"https://media.rawg.io/media/games/26d/26d4437715bee60138dab4a7c8c59c92.jpg",
   "games": [
     {
       "id": 3498,
       "slug": "grand-theft-auto-v",
```

Hard coding all of the genres so that the app does not have to call on the API so much.

```
    "name": "Grand Theft Auto V",
    "added": 19329
  },
  {
    "id": 3328,
    "slug": "the-witcher-3-wild-hunt",
    "name": "The Witcher 3: Wild Hunt",
    "added": 18504
  },
  {
    "id": 5286,
    "slug": "tomb-raider",
    "name": "Tomb Raider (2013)",
    "added": 15236
  },
  {
    "id": 4291,
    "slug": "counter-strike-global-offensive",
    "name": "Counter-Strike: Global Offensive",
    "added": 15109
  },
  {
    "id": 12020,
    "slug": "left-4-dead-2",
    "name": "Left 4 Dead 2",
    "added": 14764
  },
  {
    "id": 5679,
    "slug": "the-elder-scrolls-v-skyrim",
    "name": "The Elder Scrolls V: Skyrim",
    "added": 14615
  }
]
```

```
 },
 {
   "id": 51,
   "name": "Indie",
   "slug": "indie",
   "games_count": 52438,
   "image_background":
"https://media.rawg.io/media/games/8cc/8cce7c0e99dcc43d66c8efd42f9d03e3.jpg",
   "games": [
     {
       "id": 1030,
       "slug": "limbo",
       "name": "Limbo",
       "added": 12465
     },
     {
       "id": 3272,
       "slug": "rocket-league",
       "name": "Rocket League",
       "added": 11383
     },
     {
       "id": 422,
       "slug": "terraria",
       "name": "Terraria",
       "added": 11211
     },
     {
       "id": 9767,
       "slug": "hollow-knight",
       "name": "Hollow Knight",
       "added": 9745
     },
     {
        "id": 3612,
        "slug": "hotline-miami",
        "name": "Hotline Miami",
        "added": 9523
      },
      {
        "id": 3790,
        "slug": "outlast",
        "name": "Outlast",
        "added": 9458
      }
    ]
  },
  {
    "id": 3,
    "name": "Adventure",
    "slug": "adventure",
    "games_count": 132153,
    "image_background":
"https://media.rawg.io/media/games/7fa/7fa0b586293c5861ee32490e953a4996.jpg",
    "games": [
      {
        "id": 3498,
        "slug": "grand-theft-auto-v",
        "name": "Grand Theft Auto V",
        "added": 19329
      },
      {
        "id": 3328,
        "slug": "the-witcher-3-wild-hunt",
        "name": "The Witcher 3: Wild Hunt",
        "added": 18504
      },
      {
        "id": 5286,
        "slug": "tomb-raider",
        "name": "Tomb Raider (2013)",
        "added": 15236
      },
      {
        "id": 13536,
        "slug": "portal",
        "name": "Portal",
        "added": 14801
      },
      {
        "id": 28,
        "slug": "red-dead-redemption-2",
        "name": "Red Dead Redemption 2",
        "added": 14017
      },
      {
        "id": 3439,
        "slug": "life-is-strange-episode-1-2",
        "name": "Life is Strange",
        "added": 13951
      }
    ]
  },
  {
    "id": 5,
    "name": "RPG",
    "slug": "role-playing-games-rpg",
    "games_count": 52285,
    "image_background":
"https://media.rawg.io/media/games/d1a/d1a2e99ade53494c6330a0ed945fe823.jpg",
    "games": [
      {
```

```
    "id": 3328,
    "slug": "the-witcher-3-wild-hunt",
    "name": "The Witcher 3: Wild Hunt",
    "added": 18504
  },
  {
    "id": 5679,
    "slug": "the-elder-scrolls-v-skyrim",
    "name": "The Elder Scrolls V: Skyrim",
    "added": 14615
  },
  {
    "id": 802,
    "slug": "borderlands-2",
    "name": "Borderlands 2",
    "added": 13951
  },
  {
    "id": 58175,
    "slug": "god-of-war-2",
    "name": "God of War (2018)",
    "added": 12326
  },
  {
    "id": 3070,
    "slug": "fallout-4",
    "name": "Fallout 4",
    "added": 12265
  },
  {
    "id": 278,
    "slug": "horizon-zero-dawn",
    "name": "Horizon Zero Dawn",
    "added": 11713
```

```
      }
    ]
  },
  {
    "id": 10,
    "name": "Strategy",
    "slug": "strategy",
    "games_count": 52502,
    "image_background":
"https://media.rawg.io/media/games/fd9/fd92f105dcd6491bc5d61135033d1f19.jpg",
    "games": [
      {
        "id": 13633,
        "slug": "civilization-v",
        "name": "Sid Meier's Civilization V",
        "added": 8609
      },
      {
        "id": 10243,
        "slug": "company-of-heroes-2",
        "name": "Company of Heroes 2",
        "added": 8525
      },
      {
        "id": 13910,
        "slug": "xcom-enemy-unknown",
        "name": "XCOM: Enemy Unknown",
        "added": 7666
      },
      {
        "id": 5525,
        "slug": "brutal-legend",
        "name": "Brutal Legend",
        "added": 7590
      },
      {
        "id": 10065,
        "slug": "cities-skylines",
        "name": "Cities: Skylines",
        "added": 7454
      },
      {
        "id": 11147,
        "slug": "ark-survival-of-the-fittest",
        "name": "ARK: Survival Of The Fittest",
        "added": 7164
      }
    ]
  },
  {
    "id": 2,
    "name": "Shooter",
    "slug": "shooter",
    "games_count": 59316,
    "image_background":
"https://media.rawg.io/media/games/34b/34b1f1850a1c06fd971bc6ab3ac0ce0e.jpg",
    "games": [
      {
        "id": 4200,
        "slug": "portal-2",
        "name": "Portal 2",
        "added": 17427
      },
      {
        "id": 4291,
        "slug": "counter-strike-global-offensive",
        "name": "Counter-Strike: Global Offensive",
        "added": 15109
      },
      {
        "id": 12020,
        "slug": "left-4-dead-2",
        "name": "Left 4 Dead 2",
        "added": 14764
      },
      {
        "id": 4062,
        "slug": "bioshock-infinite",
        "name": "BioShock Infinite",
        "added": 14111
      },
      {
        "id": 802,
        "slug": "borderlands-2",
        "name": "Borderlands 2",
        "added": 13951
      },
      {
        "id": 13537,
        "slug": "half-life-2",
        "name": "Half-Life 2",
        "added": 13189
      }
    ]
  },
  {
    "id": 40,
    "name": "Casual",
    "slug": "casual",
    "games_count": 44479,
    "image_background":
"https://media.rawg.io/media/games/11f/11fd681c312c14644ab360888dba3486.jpg",
"games": [
  {
    "id": 9721,
    "slug": "garrys-mod",
    "name": "Garry's Mod",
    "added": 8741
  },
  {
    "id": 326292,
    "slug": "fall-guys",
    "name": "Fall Guys: Ultimate Knockout",
    "added": 7735
  },
  {
    "id": 9830,
    "slug": "brawlhalla",
    "name": "Brawlhalla",
    "added": 6650
  },
  {
    "id": 356714,
    "slug": "among-us",
    "name": "Among Us",
    "added": 6312
  },
  {
    "id": 1959,
    "slug": "goat-simulator",
    "name": "Goat Simulator",
    "added": 5794
  },
  {
    "id": 16343,
    "slug": "a-story-about-my-uncle",
        "name": "A Story About My Uncle",
        "added": 5459
      }
    ]
  },
  {
    "id": 14,
    "name": "Simulation",
    "slug": "simulation",
    "games_count": 65539,
    "image_background":
"https://media.rawg.io/media/games/179/179245a3693049a11a25b900ab18f8f7.jpg",
    "games": [
      {
        "id": 10035,
        "slug": "hitman",
        "name": "Hitman",
        "added": 9779
      },
      {
        "id": 654,
        "slug": "stardew-valley",
        "name": "Stardew Valley",
        "added": 8836
      },
      {
        "id": 9721,
        "slug": "garrys-mod",
        "name": "Garry's Mod",
        "added": 8741
      },
      {
        "id": 10243,
        "slug": "company-of-heroes-2",
        "name": "Company of Heroes 2",
        "added": 8525
      },
      {
        "id": 9882,
        "slug": "dont-starve-together",
        "name": "Don't Starve Together",
        "added": 8175
      },
      {
        "id": 22509,
        "slug": "minecraft",
        "name": "Minecraft",
        "added": 7524
      }
    ]
  },
  {
    "id": 7,
    "name": "Puzzle",
    "slug": "puzzle",
    "games_count": 97070,
    "image_background":
"https://media.rawg.io/media/games/948/948fe7f00b6cba8472f5ecd07a455077.jpg",
    "games": [
      {
        "id": 4200,
        "slug": "portal-2",
        "name": "Portal 2",
        "added": 17427
      },
      {
        "id": 13536,
        "slug": "portal",
      "name": "Portal",
      "added": 14801
    },
    {
      "id": 1030,
      "slug": "limbo",
      "name": "Limbo",
      "added": 12465
    },
    {
      "id": 19709,
      "slug": "half-life-2-episode-two",
      "name": "Half-Life 2: Episode Two",
      "added": 9836
    },
    {
      "id": 18080,
      "slug": "half-life",
      "name": "Half-Life",
      "added": 9056
    },
    {
      "id": 1450,
      "slug": "inside",
      "name": "INSIDE",
      "added": 7244
    }
  ]
},
{
  "id": 11,
  "name": "Arcade",
  "slug": "arcade",
  "games_count": 22552,
   "image_background":
"https://media.rawg.io/media/games/1fa/1fa75f0895240b12fc65cc98ae9649fd.jpg",
   "games": [
     {
       "id": 3612,
       "slug": "hotline-miami",
       "name": "Hotline Miami",
       "added": 9523
     },
     {
       "id": 17540,
       "slug": "injustice-gods-among-us-ultimate-edition",
       "name": "Injustice: Gods Among Us Ultimate Edition",
       "added": 8716
     },
     {
       "id": 22509,
       "slug": "minecraft",
       "name": "Minecraft",
       "added": 7524
     },
     {
       "id": 4003,
       "slug": "grid-2",
       "name": "GRID 2",
       "added": 6883
     },
     {
       "id": 3408,
       "slug": "hotline-miami-2-wrong-number",
       "name": "Hotline Miami 2: Wrong Number",
       "added": 5563
     },
     {
        "id": 16343,
        "slug": "a-story-about-my-uncle",
        "name": "A Story About My Uncle",
        "added": 5459
      }
    ]
  },
  {
    "id": 83,
    "name": "Platformer",
    "slug": "platformer",
    "games_count": 100590,
    "image_background":
"https://media.rawg.io/media/games/c89/c89ca70716080733d03724277df2c6c7.jpg",
    "games": [
      {
        "id": 1030,
        "slug": "limbo",
        "name": "Limbo",
        "added": 12465
      },
      {
        "id": 422,
        "slug": "terraria",
        "name": "Terraria",
        "added": 11211
      },
      {
        "id": 9767,
        "slug": "hollow-knight",
        "name": "Hollow Knight",
        "added": 9745
      },
      {
        "id": 41,
        "slug": "little-nightmares",
        "name": "Little Nightmares",
        "added": 9648
      },
      {
        "id": 18080,
        "slug": "half-life",
        "name": "Half-Life",
        "added": 9056
      },
      {
        "id": 3144,
        "slug": "super-meat-boy",
        "name": "Super Meat Boy",
        "added": 8678
      }
    ]
  },
  {
    "id": 59,
    "name": "Massively Multiplayer",
    "slug": "massively-multiplayer",
    "games_count": 3199,
    "image_background":
"https://media.rawg.io/media/games/1bd/1bd2657b81eb0c99338120ad444b24ff.jpg",
    "games": [
      {
        "id": 32,
        "slug": "destiny-2",
        "name": "Destiny 2",
        "added": 12304
      },
      {
```

```
      "id": 10213,
      "slug": "dota-2",
      "name": "Dota 2",
      "added": 11211
    },
    {
      "id": 766,
      "slug": "warframe",
      "name": "Warframe",
      "added": 11090
    },
    {
      "id": 290856,
      "slug": "apex-legends",
      "name": "Apex Legends",
      "added": 9854
    },
    {
      "id": 10533,
      "slug": "path-of-exile",
      "name": "Path of Exile",
      "added": 8877
    },
    {
      "id": 10142,
      "slug": "playerunknowns-battlegrounds",
      "name": "PlayerUnknown’s Battlegrounds",
      "added": 8743
    }
  ]
},
{
  "id": 1,
  "name": "Racing",
    "slug": "racing",
    "games_count": 23957,
    "image_background":
"https://media.rawg.io/media/games/ff6/ff66ce127716df74175961831ad3a23a.jpg",
    "games": [
      {
        "id": 3272,
        "slug": "rocket-league",
        "name": "Rocket League",
        "added": 11383
      },
      {
        "id": 4003,
        "slug": "grid-2",
        "name": "GRID 2",
        "added": 6883
      },
      {
        "id": 2572,
        "slug": "dirt-rally",
        "name": "DiRT Rally",
        "added": 6165
      },
      {
        "id": 58753,
        "slug": "forza-horizon-4",
        "name": "Forza Horizon 4",
        "added": 5499
      },
      {
        "id": 5578,
        "slug": "grid",
        "name": "Race Driver: Grid",
        "added": 5045
      },
      {
        "id": 4347,
        "slug": "dirt-showdown",
        "name": "DiRT Showdown",
        "added": 4373
      }
    ]
  },
  {
    "id": 15,
    "name": "Sports",
    "slug": "sports",
    "games_count": 20523,
    "image_background":
"https://media.rawg.io/media/games/640/6409857596fe6553d3bb5af9a17f6160.jpg",
    "games": [
      {
        "id": 3272,
        "slug": "rocket-league",
        "name": "Rocket League",
        "added": 11383
      },
      {
        "id": 326292,
        "slug": "fall-guys",
        "name": "Fall Guys: Ultimate Knockout",
        "added": 7735
      },
      {
        "id": 2572,
        "slug": "dirt-rally",
        "name": "DiRT Rally",
        "added": 6165
    },
    {
      "id": 53341,
      "slug": "jet-set-radio-2012",
      "name": "Jet Set Radio",
      "added": 4791
    },
    {
      "id": 9575,
      "slug": "vrchat",
      "name": "VRChat",
      "added": 4031
    },
    {
      "id": 622492,
      "slug": "forza-horizon-5",
      "name": "Forza Horizon 5",
      "added": 3991
    }
  ]
},
{
  "id": 6,
  "name": "Fighting",
  "slug": "fighting",
  "games_count": 11688,
  "image_background":
"https://media.rawg.io/media/games/91b/91be1a00c767f9e3d79353e505d7f85a.jpg",
  "games": [
    {
      "id": 17540,
      "slug": "injustice-gods-among-us-ultimate-edition",
      "name": "Injustice: Gods Among Us Ultimate Edition",
      "added": 8716
    },
    {
      "id": 108,
      "slug": "mortal-kombat-x",
      "name": "Mortal Kombat X",
      "added": 8034
    },
    {
      "id": 28179,
      "slug": "sega-mega-drive-and-genesis-classics",
      "name": "SEGA Mega Drive and Genesis Classics",
      "added": 7388
    },
    {
      "id": 9830,
      "slug": "brawlhalla",
      "name": "Brawlhalla",
      "added": 6650
    },
    {
      "id": 274480,
      "slug": "mortal-kombat-11",
      "name": "Mortal Kombat 11",
      "added": 4790
    },
    {
      "id": 44525,
      "slug": "yakuza-kiwami",
      "name": "Yakuza Kiwami",
      "added": 4105
    }
  ]
},
{
    "id": 19,
    "name": "Family",
    "slug": "family",
    "games_count": 5384,
    "image_background":
"https://media.rawg.io/media/games/3c1/3c139f67a73f0bf5ce0d8f2abf83c0d0.jpg",
    "games": [
      {
        "id": 3254,
        "slug": "journey",
        "name": "Journey",
        "added": 7876
      },
      {
        "id": 2597,
        "slug": "lego-lord-of-the-rings",
        "name": "LEGO The Lord of the Rings",
        "added": 5028
      },
      {
        "id": 3350,
        "slug": "broken-age",
        "name": "Broken Age",
        "added": 4681
      },
      {
        "id": 3729,
        "slug": "lego-the-hobbit",
        "name": "LEGO The Hobbit",
        "added": 4617
      },
      {
        "id": 1259,
        "slug": "machinarium",
        "name": "Machinarium",
        "added": 4157
      },
      {
        "id": 1140,
        "slug": "world-of-goo",
        "name": "World of Goo",
        "added": 4100
      }
    ]
  },
  {
    "id": 28,
    "name": "Board Games",
    "slug": "board-games",
    "games_count": 8302,
    "image_background":
"https://media.rawg.io/media/screenshots/220/22073cc438f0fb97967a1b53fd58c920.
jpg",
    "games": [
      {
        "id": 23557,
        "slug": "gwent-the-witcher-card-game",
        "name": "Gwent: The Witcher Card Game",
        "added": 4312
      },
      {
        "id": 327999,
        "slug": "dota-underlords",
        "name": "Dota Underlords",
        "added": 3653
      },
      {
        "id": 2055,
        "slug": "adventure-capitalist",
        "name": "AdVenture Capitalist",
        "added": 3045
      },
      {
        "id": 2306,
        "slug": "poker-night-2",
        "name": "Poker Night 2",
        "added": 1938
      },
      {
        "id": 3187,
        "slug": "armello",
        "name": "Armello",
        "added": 1835
      },
      {
        "id": 758,
        "slug": "hue",
        "name": "Hue",
        "added": 1768
      }
    ]
  },
  {
    "id": 34,
    "name": "Educational",
    "slug": "educational",
    "games_count": 15638,
    "image_background":
"https://media.rawg.io/media/games/2e8/2e8063435066d63339f137fc71a73f4e.jpg",
    "games": [
      {
        "id": 1358,
    "slug": "papers-please",
    "name": "Papers, Please",
    "added": 6180
  },
  {
    "id": 1140,
    "slug": "world-of-goo",
    "name": "World of Goo",
    "added": 4100
  },
  {
    "id": 2778,
    "slug": "surgeon-simulator-cpr",
    "name": "Surgeon Simulator",
    "added": 3596
  },
  {
    "id": 9768,
    "slug": "gameguru",
    "name": "GameGuru",
    "added": 2303
  },
  {
    "id": 13777,
    "slug": "sid-meiers-civilization-iv-colonization",
    "name": "Sid Meier's Civilization IV: Colonization",
    "added": 2135
  },
  {
    "id": 6885,
    "slug": "pirates-3",
    "name": "Sid Meier's Pirates!",
    "added": 2041
  }
```

```
    ]
  },
  {
    "id": 17,
    "name": "Card",
    "slug": "card",
    "games_count": 4488,
    "image_background":
"https://media.rawg.io/media/games/1db/1dbc3d0c9de2709e21326cdcb91468ae.jpg",
    "games": [
      {
        "id": 23557,
        "slug": "gwent-the-witcher-card-game",
        "name": "Gwent: The Witcher Card Game",
        "added": 4312
      },
      {
        "id": 28121,
        "slug": "slay-the-spire",
        "name": "Slay the Spire",
        "added": 4292
      },
      {
        "id": 18852,
        "slug": "poker-night-at-the-inventory",
        "name": "Poker Night at the Inventory",
        "added": 2561
      },
      {
        "id": 8923,
        "slug": "faeria",
        "name": "Faeria",
        "added": 2028
      },
      {
        "id": 332,
        "slug": "the-elder-scrolls-legends",
        "name": "The Elder Scrolls: Legends",
        "added": 1967
      },
      {
        "id": 2306,
        "slug": "poker-night-2",
        "name": "Poker Night 2",
        "added": 1938
      }
    ]
  }
]
```