65. Component - UpdootSection

#reactjs #frontend #component #graphql #graphql-codegen #fragment

Implement PostSnippet Fragment

- First we define a PostSnippet fragment to have graphql-codegen create us a PostSnippet type that we can use in the Updoot component
- As we saw in 22. GraphQL Mutation - login_w/ Fragments , we implement the fragments based on ObjectTypes that are already defined in the server code:
  - Post is an ObjectType defined in Post.ts entity (4. Entity - Post.ts)
- We create the following file:

/fragments/PostSnippet.graphql

```
fragment PostSnippet on Post {
  id
  createdAt
  updatedAt
  title
  textSnippet
  points
  creator {
    id
    username
  }
}
```

- We also insert this new fragment into posts.graphql to simplify it

/graphql/queries/posts.graphql

```
query Posts($limit: Int!, $cursor: String) {
  posts(limit: $limit, cursor: $cursor) {
    hasMore
    posts {
      ...PostSnippet
    }
  }
}
```

Obtain the TypeScript Type

- Run codegen to generate the TypeScript types

```
yarn gen
```

- and now we have the updated new PostSnippet type that we can use:

/generated/graphql.tsx

```
export type PostSnippetFragment = (
  { __typename?: 'Post' }
  & Pick<Post, 'id' | 'createdAt' | 'updatedAt' | 'title' | 'textSnippet' |
'points'>
  & { creator: (
    { __typename?: 'User' }
    & Pick<User, 'id' | 'username'>
  ) }
);
```

Implement UpdootSection Component

- Now we extract the Flex component that contains the buttons from the Stack component in index.ts into an Updoot component and implement the voting logic

- **Note that** in the UpdootSectionProps object we could just import the points

```
interface UpdootProps {
  points: number;
}
```

and this would be enough for this particular implementation, however we want to be able to access all fields of a post, so that if in the future this component needs to access more than the points, it will be able to do that.

- For this, we could import a PostsQuery object as follows, and it would contain all fields of a post at the deepest level

```
import { PostsQuery } from "../generated/graphql";

interface UpdootProps {
```

```
  post: PostsQuery["posts"]["posts"][0];
}
```

- Instead we simply use the PostSnippet type that we obtained
- We also use the useVoteMutation() hook we created with graphql-codegen
- **Note that** here, instead of using the fetching state we could obtain with `[{fetching}, vote] = useVotingMutation()` we instead implement the loading indicators on the buttons separately, since the fetching variable would not allow us to know which button to display as "loading". In fact the operation variable shows us what value was passed to the vote() function, but that is updated only after the function is completed so it is also not useful for us

/components/UpdootSection.tsx

```
import { ChevronUpIcon, ChevronDownIcon } from "@chakra-ui/icons";
import { Flex, IconButton } from "@chakra-ui/react";
import React, { useState } from "react";
import { PostSnippetFragment, useVoteMutation } from "../generated/graphql";

interface UpdootSectionProps {
  post: PostSnippetFragment;
}

export const UpdootSection: React.FC<UpdootSectionProps> = ({ post }) => {
  const [loadingState, setLoadingState] = useState<
    "updoot-loading" | "downdoot-loading" | "not-loading"
  >("not-loading");
  const [, vote] = useVoteMutation();
  return (
    <Flex direction="column" justifyContent="center" alignItems="center" mr={4}>
      <IconButton
        onClick={async () => {
          setLoadingState("updoot-loading");
          await vote({
            postId: post.id,
            value: 1,
          });
          setLoadingState("not-loading");
        }}
        isLoading={loadingState === "updoot-loading"}
        boxSize={6}
```

```
        icon={<ChevronUpIcon />}
        aria-label={"Upvote post"}
      />
      {post.points}
      <IconButton
        onClick={async () => {
          setLoadingState("downdoot-loading");
          await vote({
            postId: post.id,
            value: -1,
          });
          setLoadingState("not-loading");
        }}
        isLoading={loadingState === "downdoot-loading"}
        boxSize={6}
        icon={<ChevronDownIcon />}
        aria-label={"Downvote post"}
      />
    </Flex>
  );
};
```