BeerToBeer

Group Members:

- Alfani Giacomo
- Di Stefano Roberto
- Paciotti Marta
- Pastorelli Matteo

Public website can be found here, gitHub repository here.

Introduction:

BeerToBeer is an application that allows the user to discover the world of artisan beers made by Brewdog's, a young scottish brewery founded in 2007 by a couple of beer enthusiasts (https://www.brewdog.com/uk).
Inside it, every user can search for Brewdog's artisan beers, see the characteristics of them, read the user's reviews about them, write one on his own and reply to others, and discover Brewdog's pubs all over UK.
The goal of the application is spreading the knowledge about the world of artisan beers, still poorly known by most people, and, meanwhile, sharing user experience between each other.
The user is someone that knows already something about this world but also someone that is curious and wants to know something about it.

Technologies:

To build BeerToBeer the following technologies have been used:

- **React JS**: a JS library used to build the whole application;
- **JavaScript**: used to create functions, hooks and props and to communicate with the persistence layer;
- **MUI**: a simple, customizable, and accessible library of React components;
- **Bulma**: a CSS framework;
- **Redux Toolkit**: to store and maintain application global state;
- **HTML** and **CSS**: used inside JSX components;
- **Firebase**: a Google open source platform that has been used as a realtime cloud database for storing users and their informations (like login info and profile image), their reviews and replies, the number of times the page of each beer has been visited (used display the most popular ones in the homepage), the number of likes for each beer, the list of favourite beers for each registered user, Brewdog's pubs with all their characteristics (photo, description and location);
- **Firebase Hosting**: used to host the application on a public web server;

- **Punk API**: was used to retrieve the characteristics of each beer, its description and its image (https://punkapi.com);
- **React-leaflet**: an open-source JavaScript library for interactive maps.

Logical organization and directory structure:

The project directories are structured as follows:

- **Components**: all the React components: some of them are simple components, some others are instead containers, that is they have the responsability to contain the simple components and providing them data via queries, props, Redux store etc;
- **Context**: contains context for users authentication available for all the app components;
- **Hooks**: some custom useful hooks;
- **Pages**: contains all the application pages;
- **Services**: contains several directories, each one of them for a different service (better explained later);
  and a Javascript file (persistence-manager.js) to avoid direct coupling between files responsible to manage persistence and the other ones;
- **Slices**: contains the Redux store slices;
- **Store**: a file used for combining slices that contribute to the overall state;
- **Style**: contains a file with the selected color palette for the application.

Service directory contains the following files:

- **Firebase/conf-firebase.js:** maintains the Firebase API configuration**;**
- **Firebase/firestore-database/crud-op.js:** responsible for the interaction with Firestore (a no-sql database included in firebase stack);
- **Firebase/storage/manage_storage.js:** responsible for the interaction with Firestore (a cloud service useful to store images and video);
- **BeerApi/BeerApiHeandler.js:** responsible for the interaction with Punk API;
- **Persistence_manager.js:** an interface between persistence layer (Firebase and API) and the others above;
- **Utility/Review_utility**: some utility functions.

Functionality:

- Search beer by full name/partial name;
- Display list of beers obtained from the search;
- Filtering and sorting the search results for some beers;
- View all the characteristics of a single beer: Name, image, ABV (i.e. alchool percetage), IBU (i.e. International Bitterness Unit), SRM (i.e. Standard Reference Method), description and food pairing;
- Like a beer;
- Beers can be added to user’s favorites list;
- Insertion of a beer review or reply to others' reviews and delete them;
- Reading all the reviews for a single beer;
- Search pub by full name/partial name;

- View pubs in the UK by scrolling through their list or on the map;
- View all the characteristics of a single pub: name and description;
- Insertion of a pub review or replay to others' reviews and delete them;
- Reading all the reviews for a single pub;
- View your favorites list;
- Change profile picture and username;
- Admin can change slider images and default profile image for users and delete user's reviews.

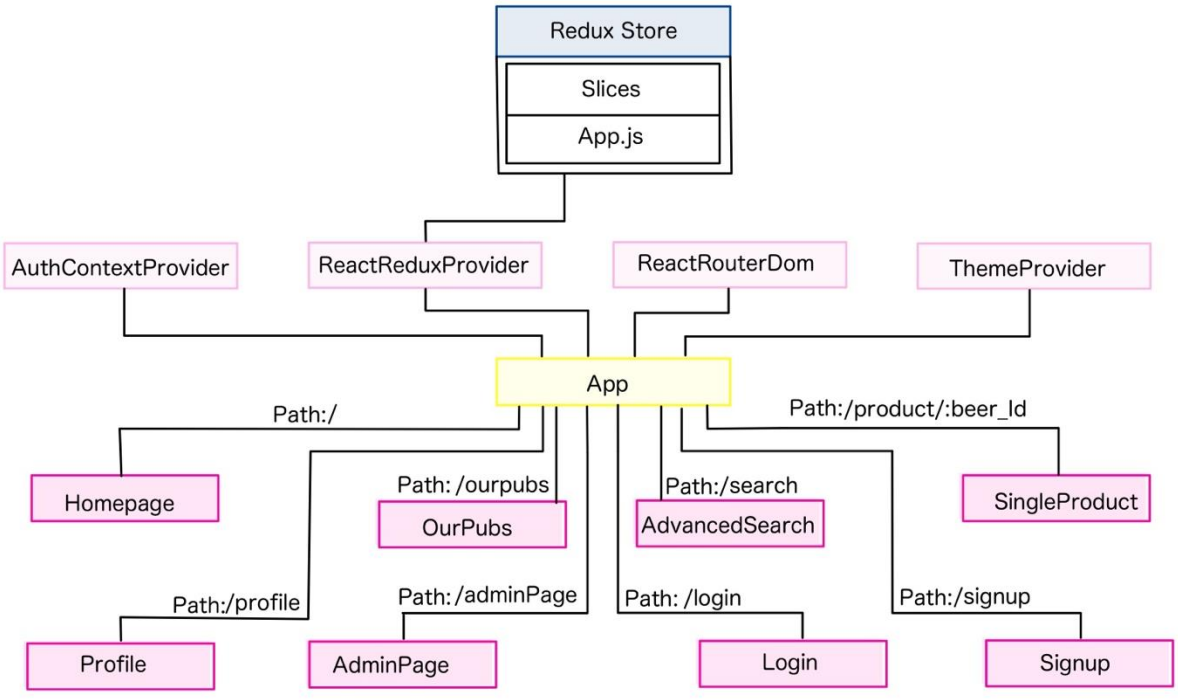
Use Case:

A user that experienced a Brewdog's beer, decides to visit the application to review that beer: he searches the name in the home search bar, and he lands on the product page in which he can write his own review. Meanwhile he can also read, in that page, other people's opinions about that beer and reply to them, or he can explore related products, by clicking on their image, and adding beers to his favorite list. Finally he can decide to explore "our pub" page where he can discover official Brewdog's pub position in UK.

App structure:

In this section is shown the structure of the whole application, starting from the general one, with the providers and the first level children of App, followed by the structure of every single page and its components, highlighthing handlers functions and general and local states.

General structure:



App as the following four providers:

- **AuthContextProvider**: manages user's authorization inside the app. The app distinguishes between three different types of users:
    - *Unlogged user*: can visit the app, read other user's reviews about beers and pubs;
    - *Logged user*: can do everything the unlogged can, but also write reviews about beers and pubs and reply to others' ones or delete his own reviews and replies, can add beers to favorites list (that can be seen in the profile page) and leave likes to them. In profile page he can set and change username and profile photo;
    - *Admin*: can do everything the logged can do, but also delete everyone's reviews and replies, change homepage's slider photos and user's default profile photo.
- **React-ReduxProvider**: creates communication between App and Redux Store, composed by two parts:
    - Slices directory, containings only the store slices;
    - App.js: in which the store is created.
- **ReactRouterDom**: manages app navigation (better explained in the next paragraph);
- **ThemeProvider**: provides app palette's colors to all components, as shown in the following picture:

```
palette: {
    primary: {
        light: '#4fc3f7',
        main: '#29b6f6',
        dark: '#0288d1',
        contrastText: '#fff',
    },
    secondary: {
        light: '#29b6f6',
        main: '#0288d1',
        dark: '#046194',
        contrastText: '#fff'
    },
    error: {
        light: '#ff7961',
        main: '#f44336',
        dark: '#ba000d',
        contrastText: '#000',
    },
    warning: {
        light: '#facf44',
        main: '#f5a844',
        dark: '#e65100',
        contrastText: '#000',
    },
    success: {
        light: '#4caf50',
        main: '#2e7d32',
        dark: '#1b5e20',
        contrastText: '#fff'
    },
    info: {
        light: '#f8f8f8',
        main: '#c0bfbf',
        dark: '#333333',
        contrastText: '#212121'
    },
},
```

ReactRouterDOM and User Authentication:

To handle user managment we used User Authentication Firebase Service, that makes available high level API to register, log users to the web application with all the Front-End security precautions (jwt etc). Current User, and all the functions to manage his authentication, are stored in the AuthContext, so that all these informations can be shared to the application components (in most cases these are used to achieve a more customizable page, i.e. adding some features that only a logged user or admin user can benefit).

To navigate to the app pages, we decided to take the most famous library in the React community: React-router-dom (v6), as shown in the following picture:

```
function App() {
    return (
        <Provider store={store}>
            <ThemeProvider theme={theme}>
                <BrowserRouter>
                    <Routes>
                        <Route path="/" element={< Homepage />}/>
                        <Route element={<PrivateRoute />}>
                            <Route path="/profile" element={< Profile />}/>
                        </Route>
                        <Route path="/login" element={<Login/>} />
                        <Route path="/signup" element={<Signup/>} />
                        <Route path="/search/:searchTerm?" element={<Search/>} />
                        <Route path="/ourpubs" element={<OurPubs/>} />
                        <Route path="/product/:beer_Id?" element={<SingleProductPage/>} />
                        <Route element={<AdminPrivateRoute />}>
                            <Route path="/adminPage" element={< AdminPage />}/>
                        </Route>
                        <Route path="*" element={<NotFound code = "404 Not Found"
                                                           message="The requested page could not be found."/>} />
                    </Routes>
                </BrowserRouter>
            </ThemeProvider>
        </Provider>
    );
}

1+ usages  Matteopast01
export default App;
```

To add a new route to the application, a Route component must be inserted into the App component with two props: the url and the element that must be rendered. At first glance, watching the previous image, it is not easy to grasp how to handle private Routes. Indeed, we did not find an already built-in solution in the library to make a route available only to a particular kind of user, so we built a Private Route component:

```
export const PrivateRoute = () => {
    const {currentUser} = useContext(AuthContext);

    return ( !!currentUser ? <Outlet />  : <Navigate to="/login" />);
};
```

Its work is, simply, to show the right component to the user according to his privilege (provided by the Auth Context): if the user is logged (not null by a technical point of view) the specified page is available (wrapped by the Outlet component), otherwise he is rendered to the login page.
For example, Profile page can be exclusively seen by logged users, but how did we reach that goal?

```
<Route element={<PrivateRoute />}>
    <Route path="/profile" element={< Profile />}/>
</Route>
```

As shown in the picture above, we passed to Route component the private Route component and the component that must be private (retrieved calling the Outlet component of the library). This mechanism can be iterated to all the private pages of the application as shown in the previous image.

Page structure:

App has the following component hierarchy:

Home page:

- Header:
  - CustomAccountButton;
  - CustomButton x3;
  - SearchBar;
- SimpleSlider;
- BeerContainer:
  - DropDown;
  - CardList:
    - CustomCard:
      - BeerCardDescription:
        - CustomIconButton x2;
- Footer.

OurPubs:

- Header:
  - CustomAccountButton;
  - CustomButton x3;
  - SearchBar;
- PageSwitch:
  - PubContainer / SinglePub:
    - PubContainer:
      - CardList:
        - CustomCard x3;
    - SinglePub:
      - CustomIconButton;
      - CustomCard;
      - InputReview:
        - CustomButton;
      - Option:
        - CustomButton x2;
      - PubReviewContainer:
        - Review (as many as review written):
          - CustomButton;

- Maps;
- Footer.

Advanced Search:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
    - SearchBar;
- AdvancedSearch;
- Sorting:
    - DropDown x2;
- ResultContainer:
    - CardList:
        - CustomCard:
            - BeerCardDescription:
                - CustomIconButton;
- Footer.

SingleProduct:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
    - SearchBar.
- CustomCard:
    - ProductCardDescription:
        - CustomButton x2;
- InputReview:
    - CustomButton;
- Option:
    - CustomButton x2;
- ProductReviewContainer:
    - Review (as many as review written):
        - CustomButton;
- Footer.

Profile:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
    - SearchBar;
- CustomCard:
    - Popup;
        - CustomIconButton;
        - CustomButton;
        - CustomButton;
        - CustomButton:
            - Option:
                - CustomButton x2;

            - ImagesUploader:
                - FileUploadButton;
- FavoritesContainer:
    - CardList:
        - CustomCard:
            - BeerCardDescription:
                - CustomIconButton x2;
- Footer.

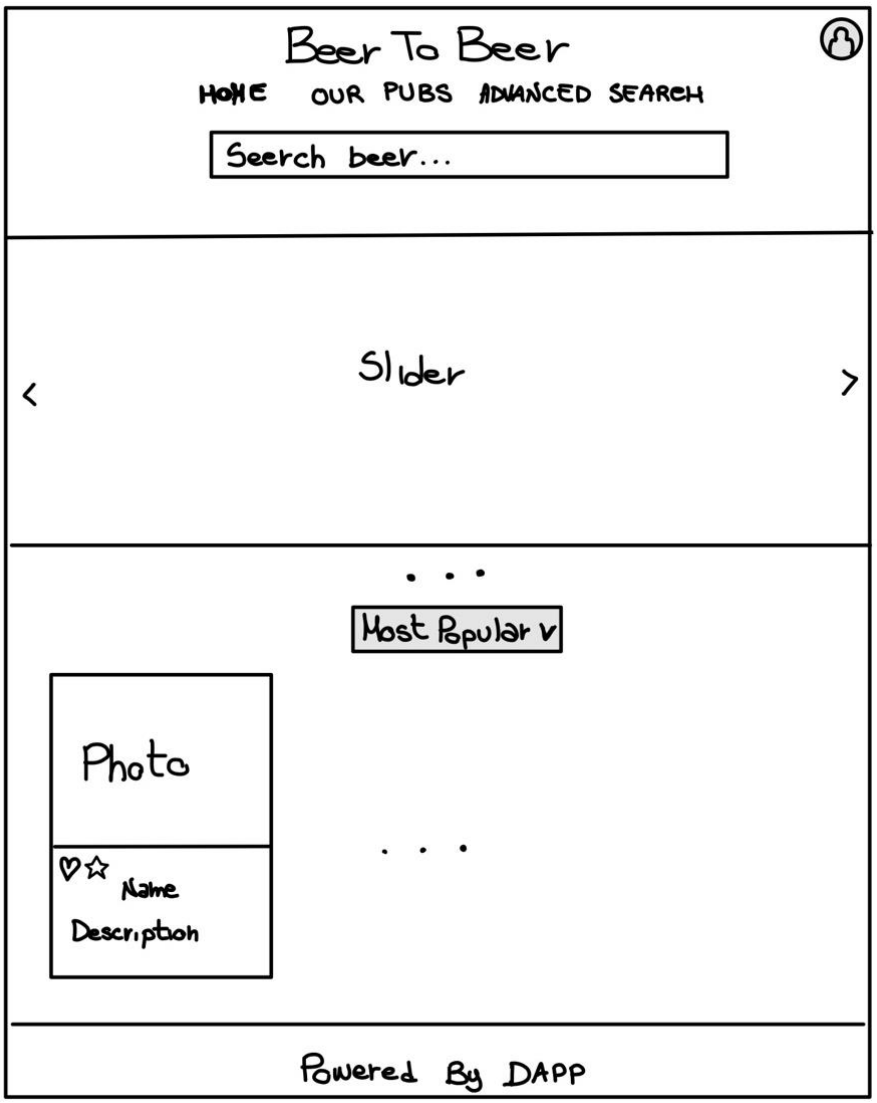
AdminPage:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
- ImagesUploader x2:
    - FileUploadButton;
- Footer.

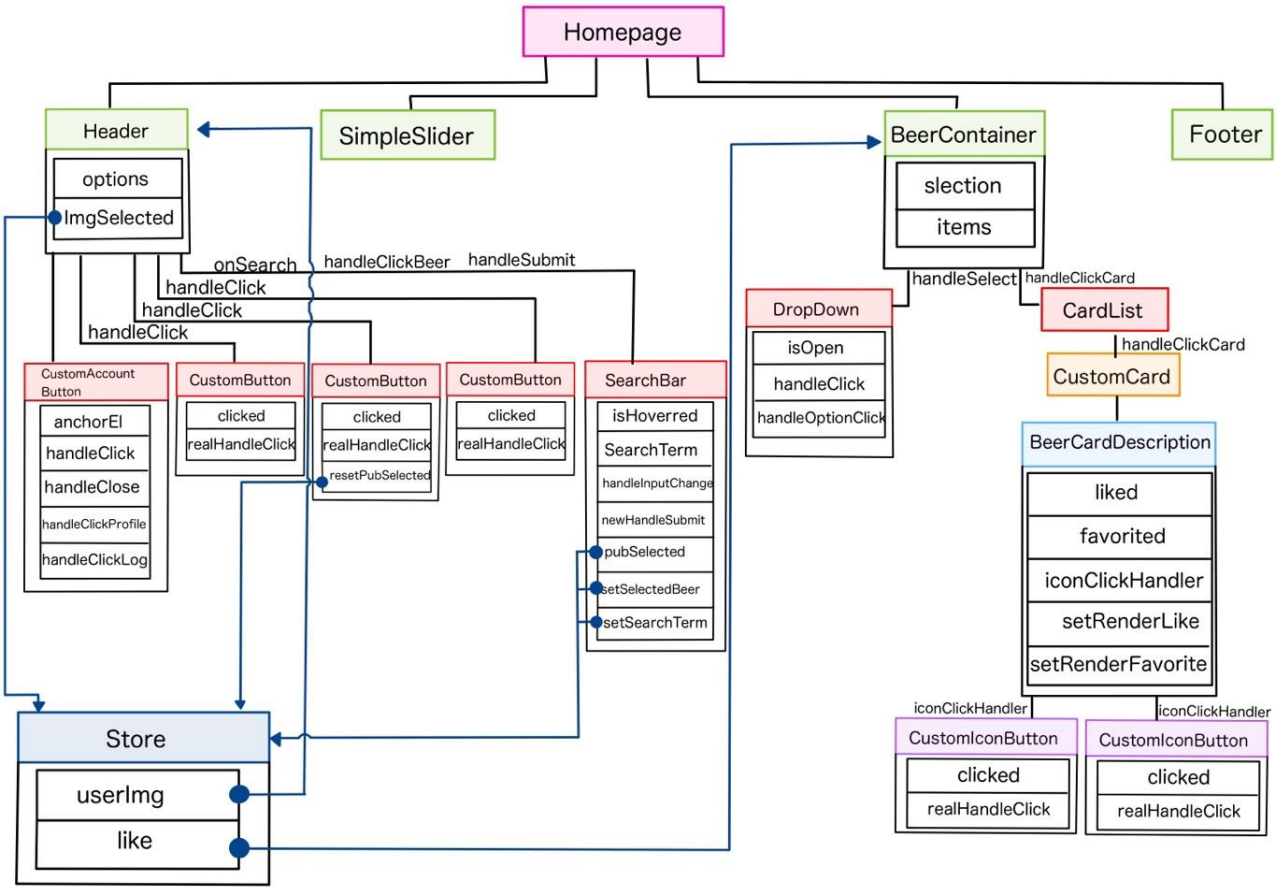
Login:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
- CustomButton;
- Footer.

Signup:
- Header:
    - CustomAccountButton;
    - CustomButton x3;
- CustomButton;
- Footer.

Homepage:

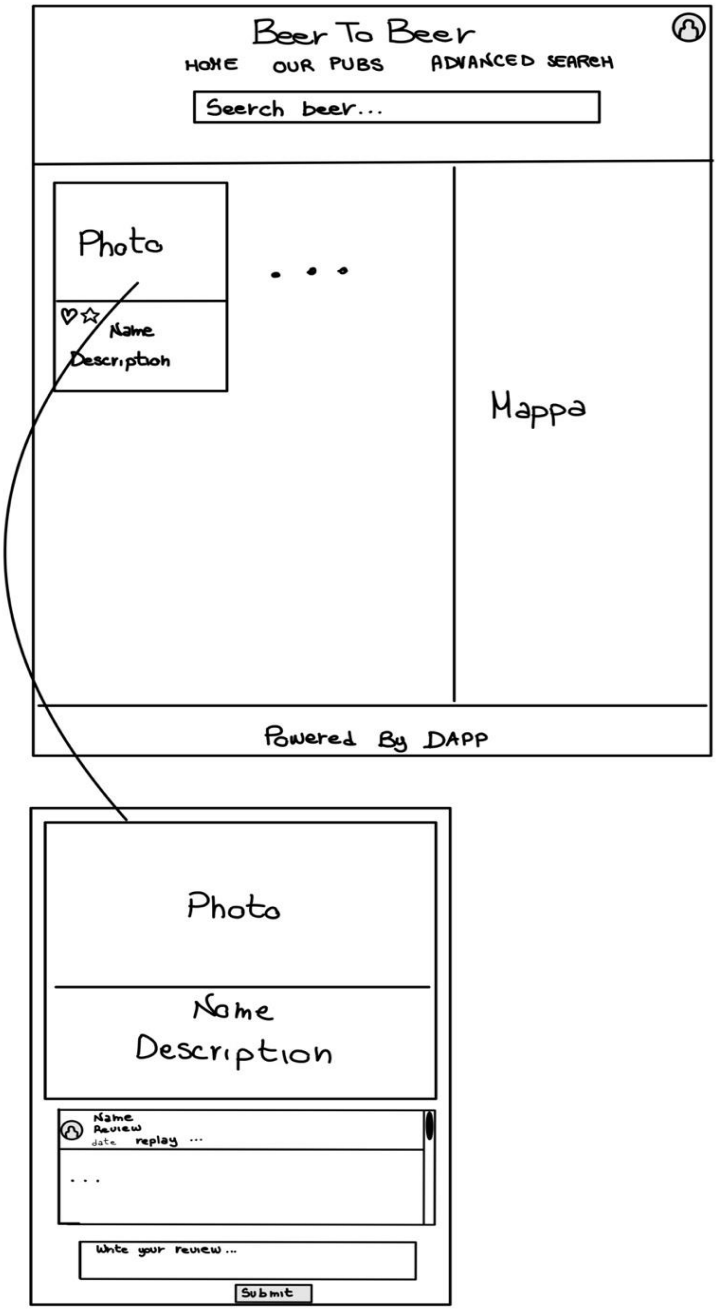
Homepage is the BeerToBeer initial page and its primary goal is to welcome the user in Brewdog's artisan beers world. The main focus is given to the big search bar in the upper part of the page, so that the user is invited to search for his favorite beers, or, otherwise, navigate the application through the navigation buttons.
The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

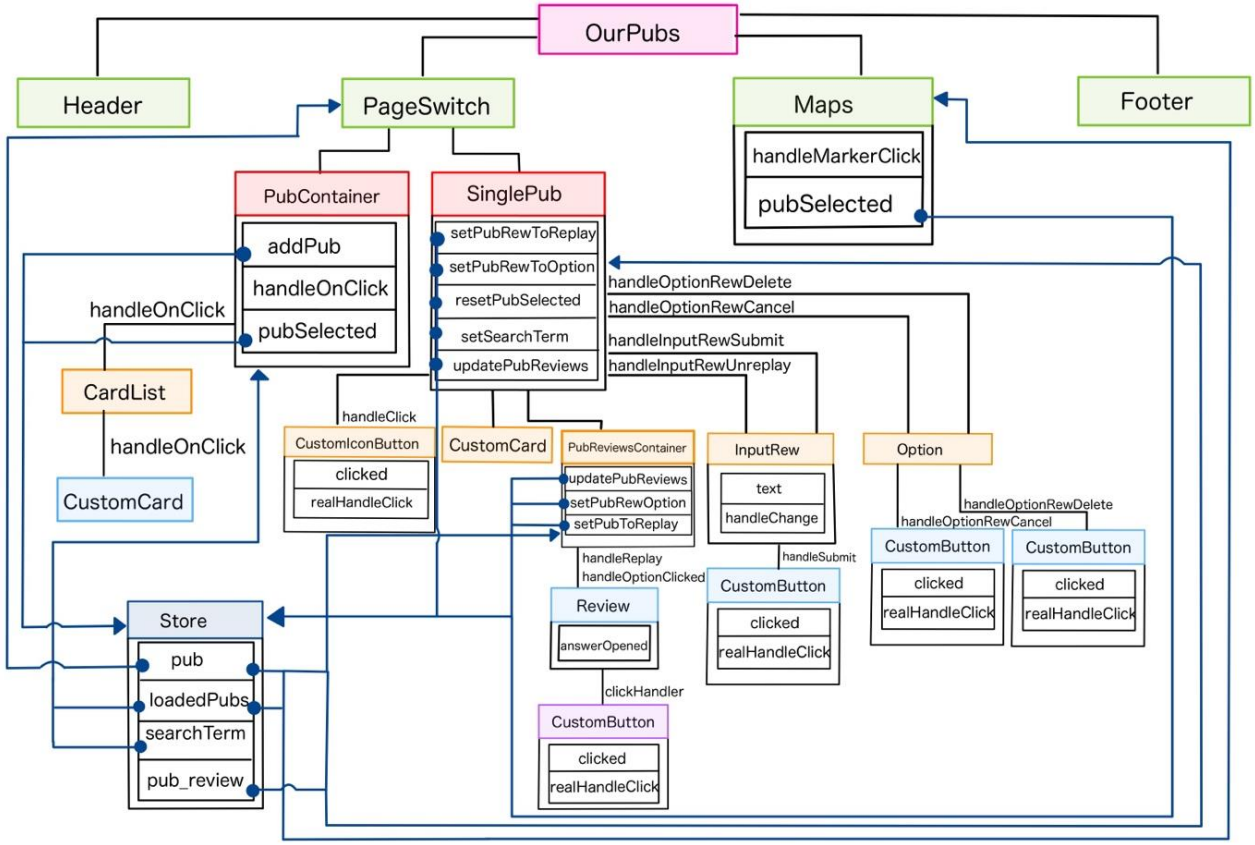



Homepage
Header
options
ImgSelected
SimpleSlider
BeerContainer
slection
items
Footer
onSearch
handleClickBeer
handleSubmit
handleClick
handleClick
handleClick
handleSelect
handleClickCard
DropDown
isOpen
handleClick
handleOptionClick
CardList
handleClickCard
CustomCard
CustomAccount Button
anchorEl
handleClick
handleClose
handleClickProfile
handleClickLog
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
resetPubSelected
CustomButton
clicked
realHandleClick
SearchBar
isHoverred
SearchTerm
handleInputChange
newHandleSubmit
pubSelected
setSelectedBeer
setSearchTerm
BeerCardDescription
liked
favorited
iconClickHandler
setRenderLike
setRenderFavorite
iconClickHandler
iconClickHandler
CustomIconButton
clicked
realHandleClick
CustomIconButton
clicked
realHandleClick
Store
userImg
like

Our Pubs:

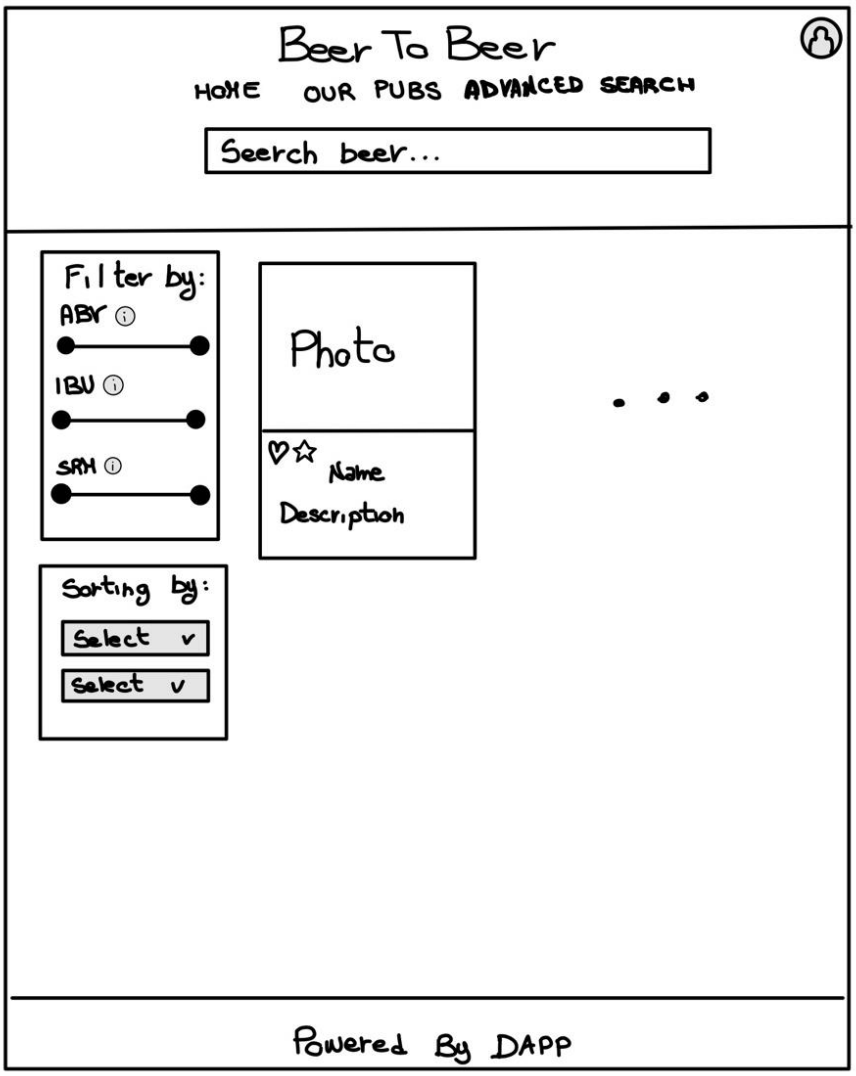
In this page the user can see and find the locations of all Brewdog's pubs all over UK and see the reviews about them and, if logged, leave one of their own and comment someone else's ones. On the left side of the page is displayed the list of all the pubs or a single pub, if selected with a click, and on the right side the map with a marker for each of them.
The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

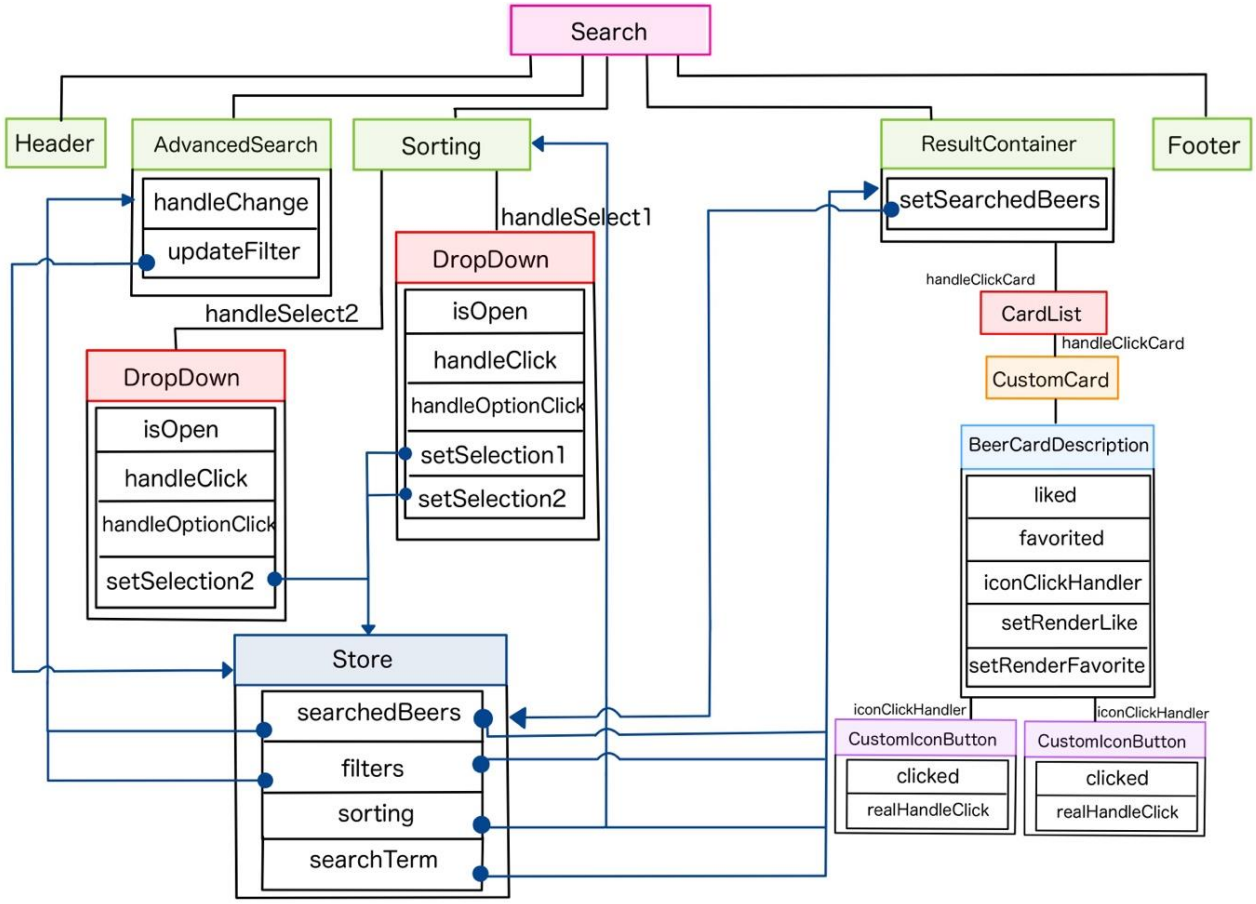



OurPubs
Header
PageSwitch
Maps
Footer
handleMarkerClick
pubSelected
PubContainer
addPub
handleOnClick
pubSelected
SinglePub
setPubRewToReplay
setPubRewToOption
resetPubSelected
setSearchTerm
updatePubReviews
handleOptionRewDelete
handleOptionRewCancel
handleInputRewSubmit
handleInputRewUnreplay
handleOnClick
CardList
handleOnClick
CustomCard
handleClick
CustomIconButton
clicked
realHandleClick
CustomCard
PubReviewsContainer
updatePubReviews
setPubRewOption
setPubToReplay
InputRew
text
handleChange
Option
handleOptionRewDelete
handleOptionRewCancel
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
handleReplay
handleOptionClicked
handleSubmit
CustomButton
clicked
realHandleClick
Review
answerOpened
clickHandler
CustomButton
clicked
realHandleClick
Store
pub
loadedPubs
searchTerm
pub_review

Advanced Search:

This page can be used by the users to make a more accurate search for all Brewdog's beers and filter the resulting list by ABV, IBU, SRM, that is the main characteristics of a beer, and sorting them by IBU, alphabetical order or no order at all.
The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

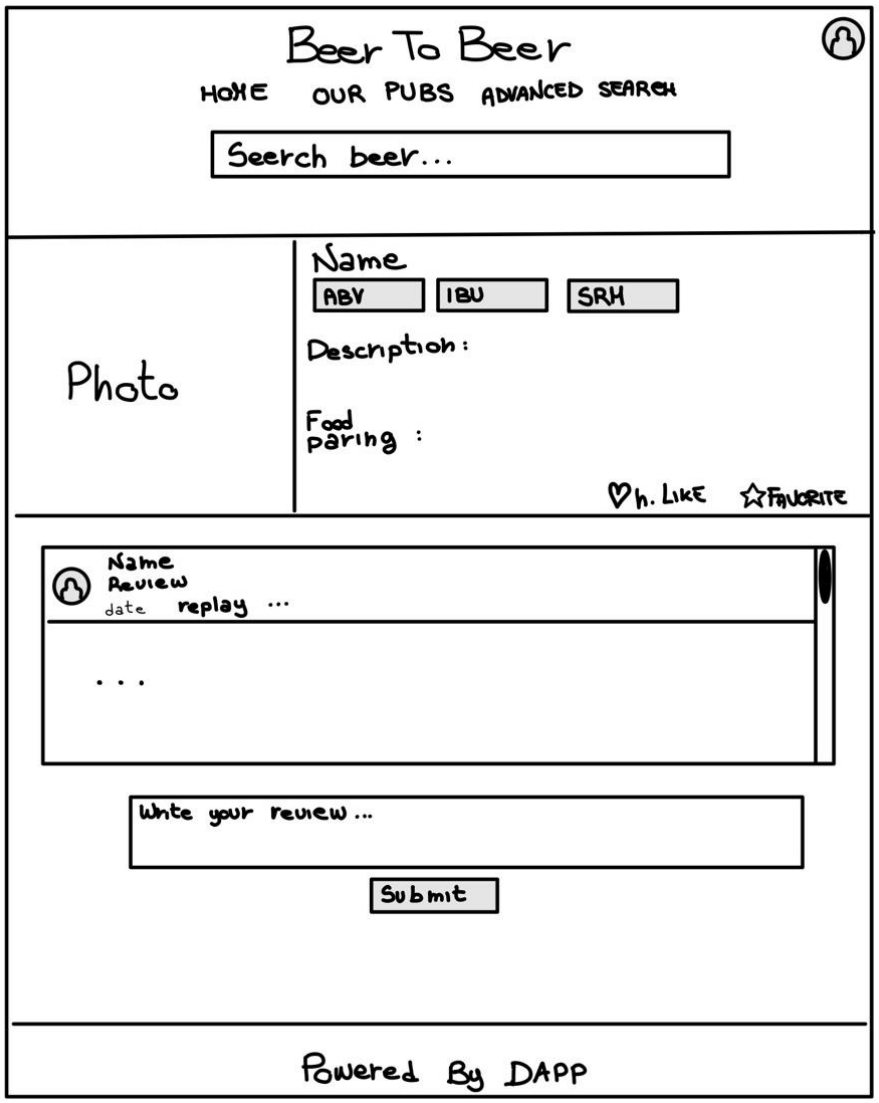


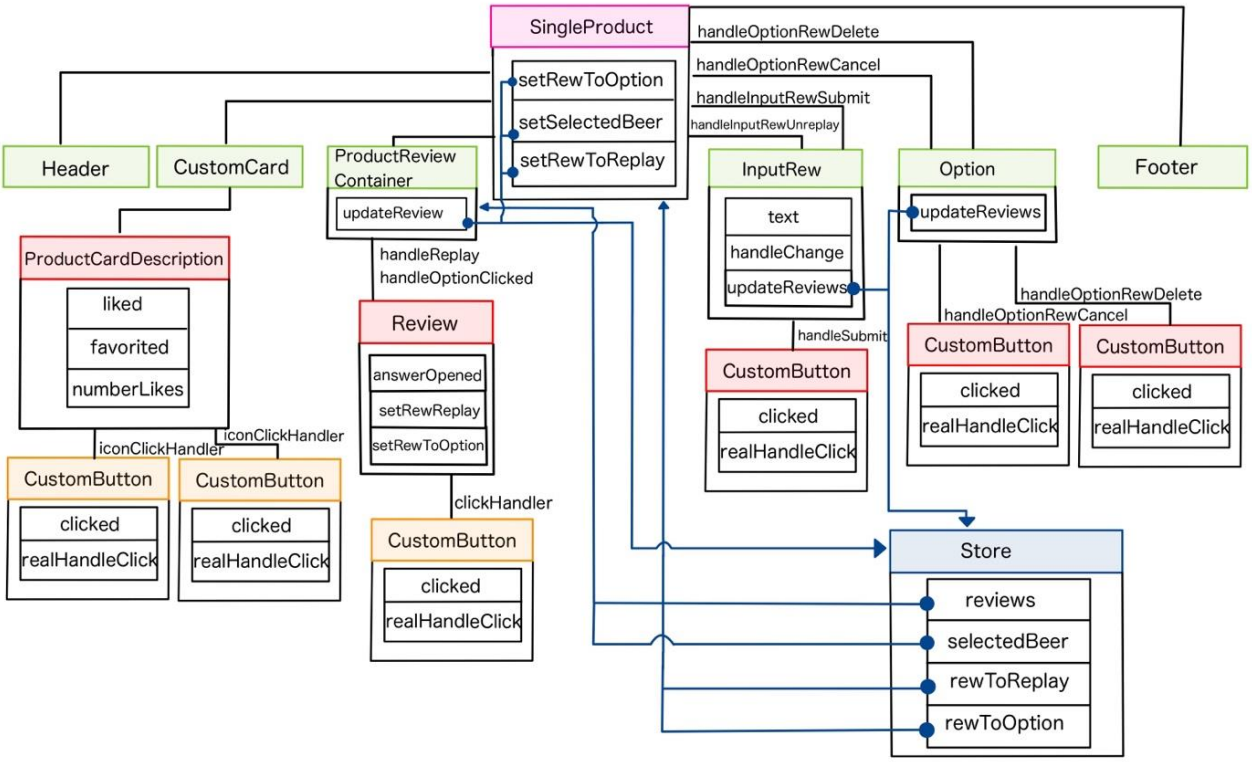

Search
Header
AdvancedSearch
handleChange
updateFilter
Sorting
handleSelect1
DropDown
isOpen
handleClick
handleOptionClick
setSelection1
setSelection2
handleSelect2
DropDown
isOpen
handleClick
handleOptionClick
setSelection2
ResultContainer
setSearchedBeers
Footer
handleClickCard
CardList
handleClickCard
CustomCard
BeerCardDescription
liked
favorited
iconClickHandler
setRenderLike
setRenderFavorite
Store
searchedBeers
filters
sorting
searchTerm
iconClickHandler
CustomIconButton
clicked
realHandleClick
iconClickHandler
CustomIconButton
clicked
realHandleClick

Single product:

In this page the user can find the product he clicked on, with all its characteristics, a photo, its full description and other user's reviews. Here he can also leave a review or reply to other user's ones.

The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

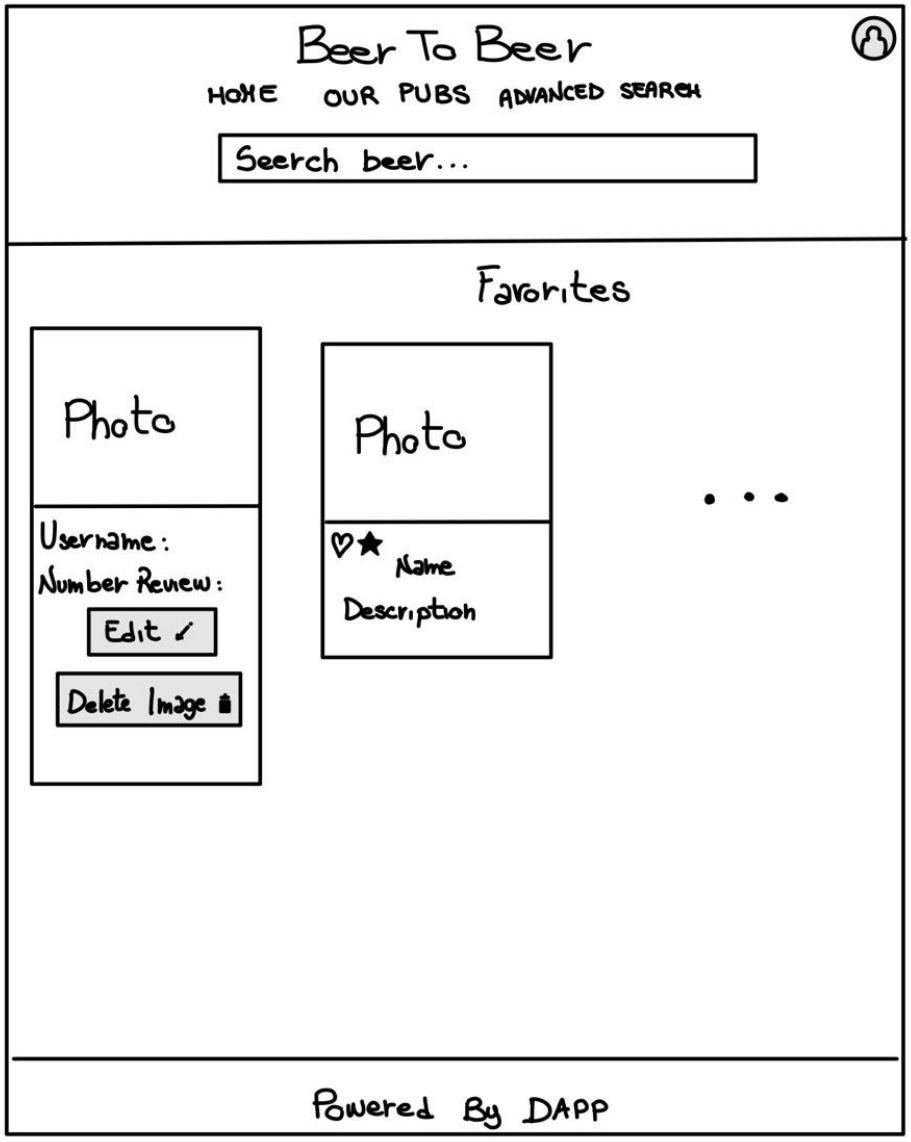


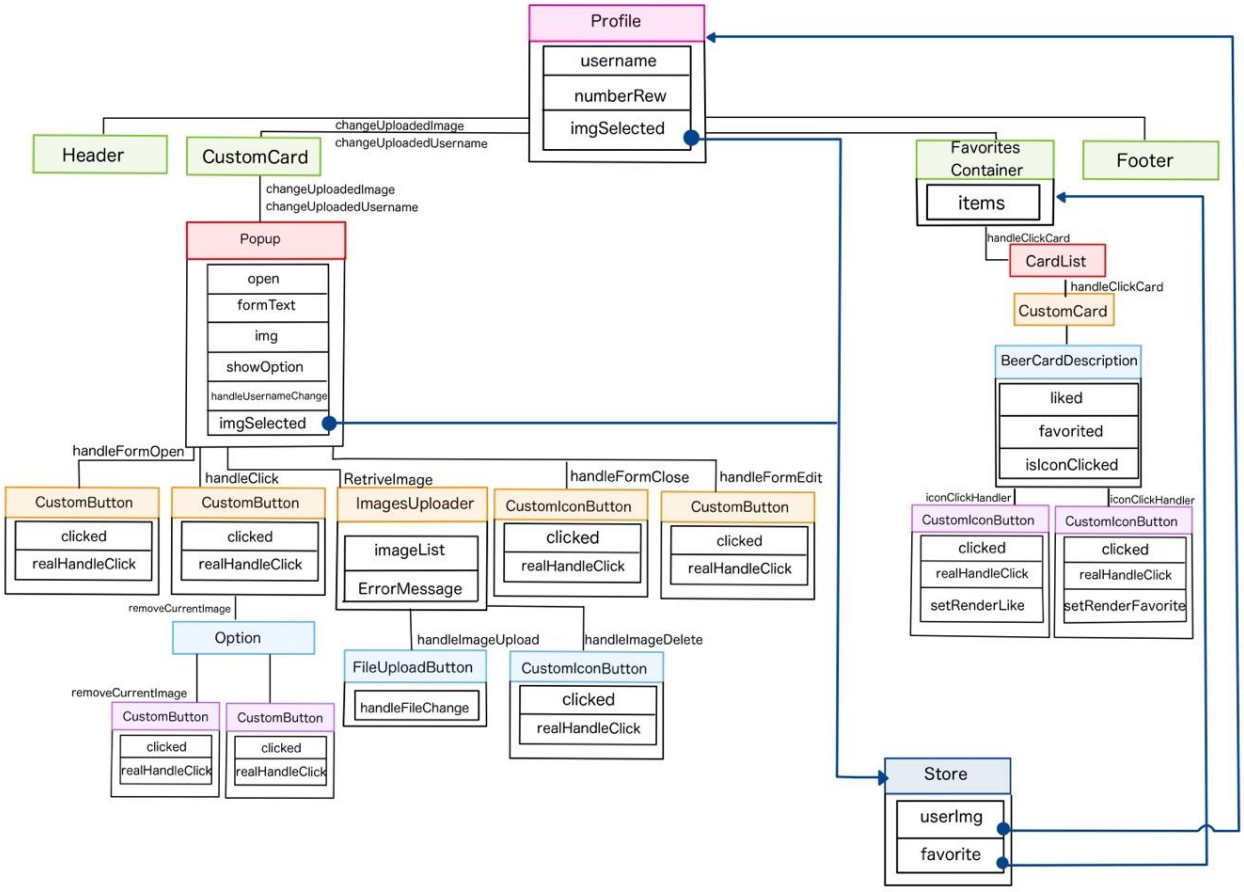

SingleProduct
setRewToOption
setSelectedBeer
setRewToReplay
handleOptionRewDelete
handleOptionRewCancel
handleInputRewSubmit
handleInputRewUnreplay
Header
CustomCard
ProductReview Container
updateReview
InputRew
text
handleChange
updateReviews
Option
updateReviews
Footer
ProductCardDescription
liked
favorited
numberLikes
handleReplay
handleOptionClicked
Review
answerOpened
setRewReplay
setRewToOption
handleSubmit
CustomButton
clicked
realHandleClick
handleOptionRewDelete
handleOptionRewCancel
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
iconClickHandler
iconClickHandler
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
clickHandler
CustomButton
clicked
realHandleClick
Store
reviews
selectedBeer
rewToReplay
rewToOption

Profile:

This is the registered user's profile page, in which he can set and modify his avatar, see the number of reviews he wrote about beers on the application and a list of favorite beers he inserted here by clicking on the star icon in any beer card found on the application.

The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

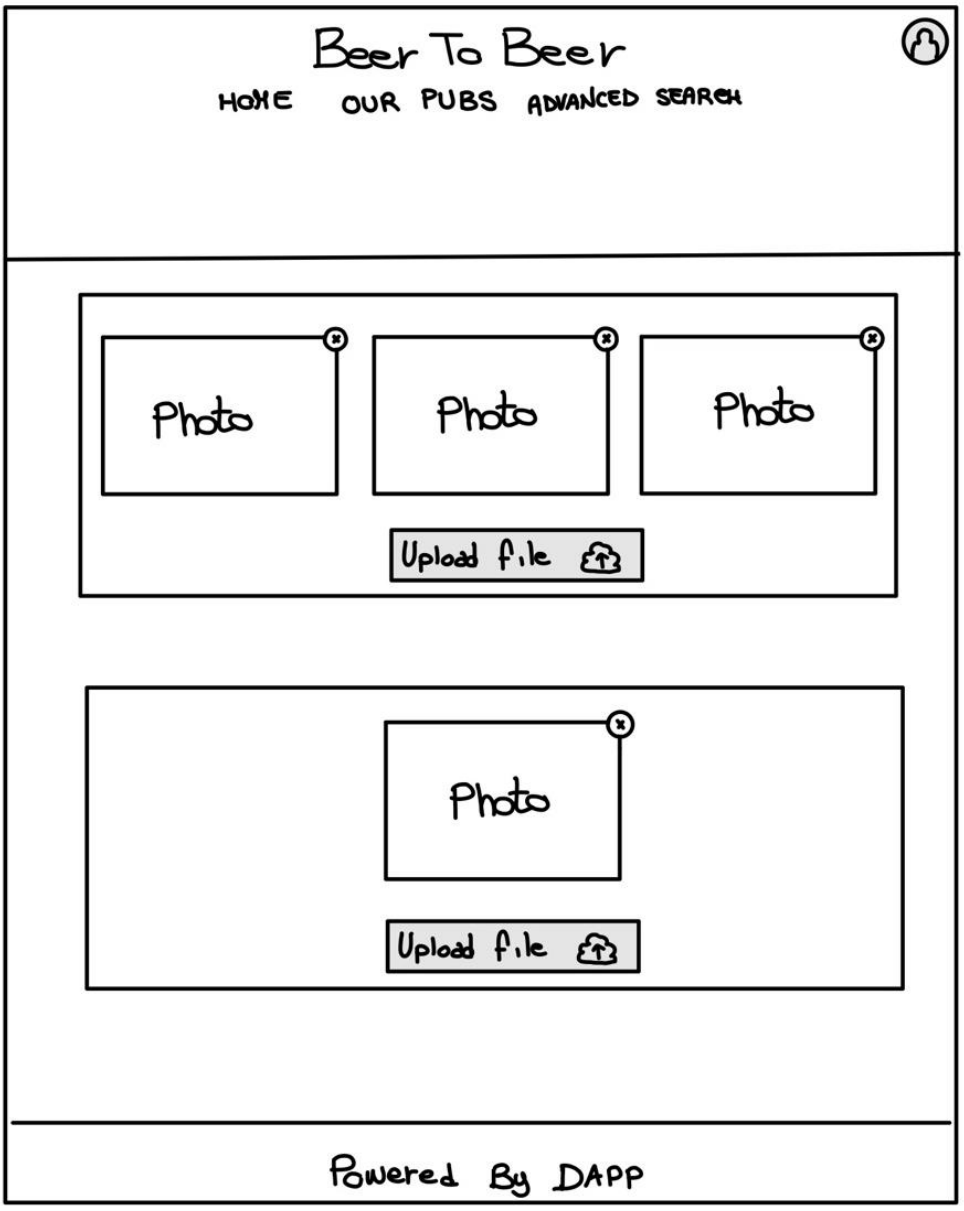


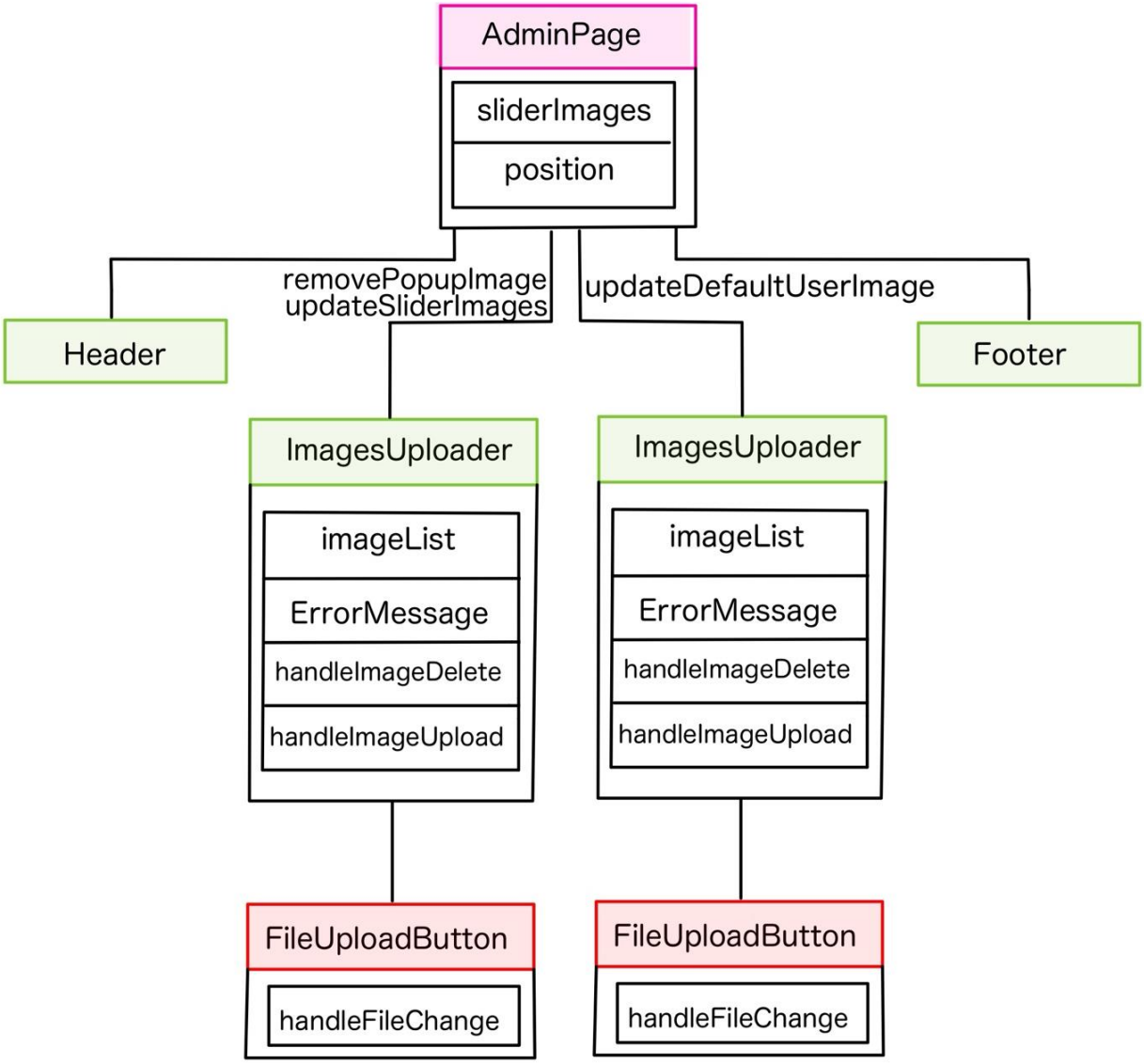

Profile
username
numberRew
imgSelected
Header
CustomCard
changeUploadedImage
changeUploadedUsername
Favorites Container
items
Footer
changeUploadedImage
changeUploadedUsername
Popup
open
formText
img
showOption
handleUsernameChange
imgSelected
handleClickCard
CardList
handleClickCard
CustomCard
BeerCardDescription
liked
favorited
isIconClicked
handleFormOpen
handleClick
RetriveImage
handleFormClose
handleFormEdit
iconClickHandler
iconClickHandler
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
ImagesUploader
imageList
ErrorMessage
CustomIconButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
CustomIconButton
clicked
realHandleClick
setRenderLike
CustomIconButton
clicked
realHandleClick
setRenderFavorite
removeCurrentImage
Option
handleImageUpload
handleImageDelete
removeCurrentImage
FileUploadButton
handleFileChange
CustomIconButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
CustomButton
clicked
realHandleClick
Store
userImg
favorite

AdminPage:

A dashboard page in which the administrator can change the images of the home slider and the user default profile avatar.

The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

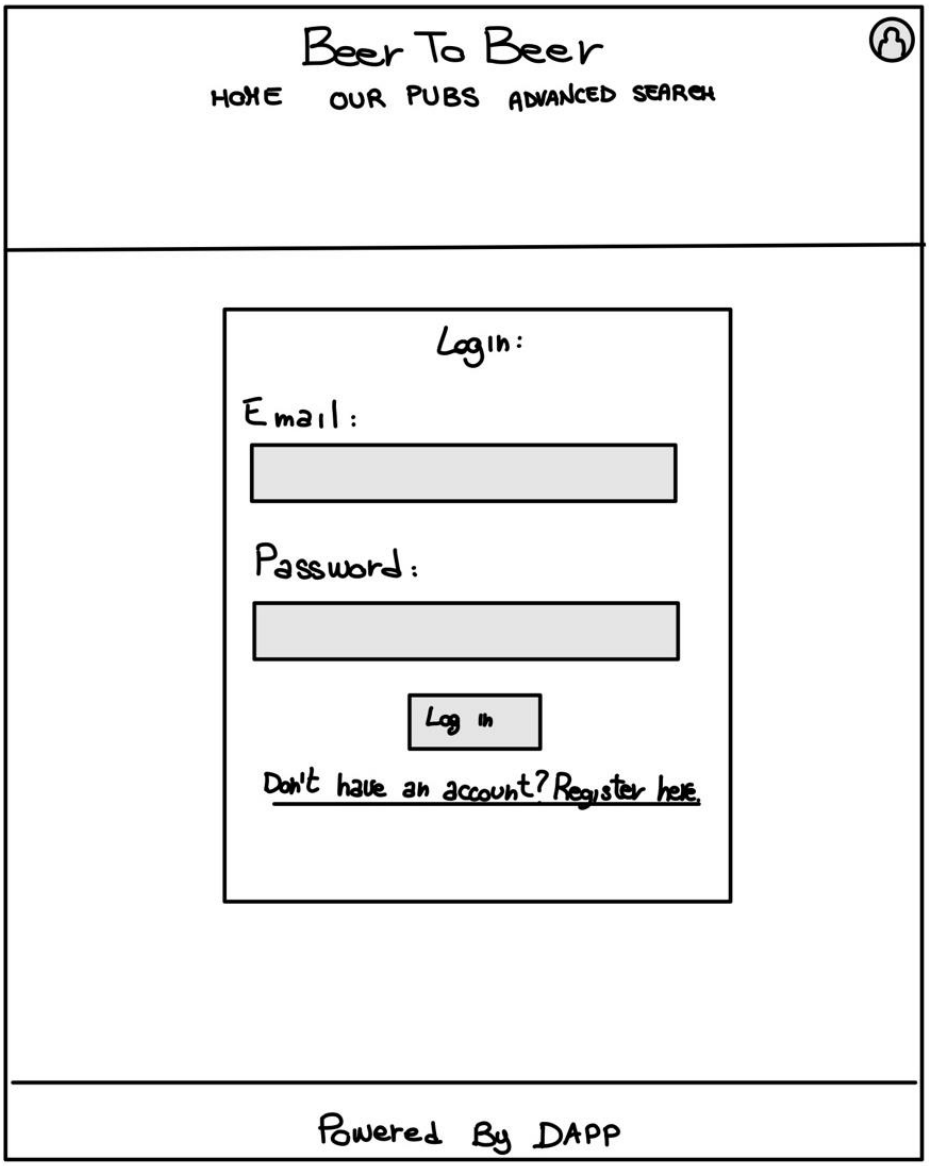


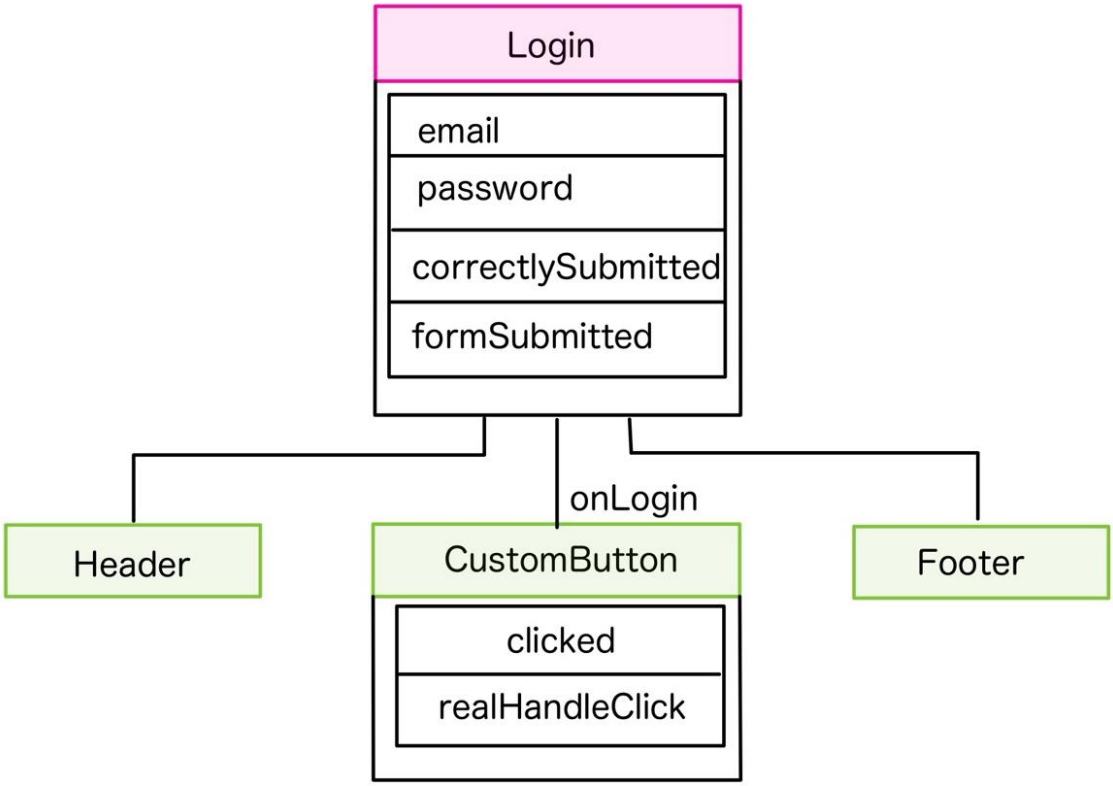

AdminPage
sliderImages
position
removePopupImage
updateSliderImages
updateDefaultUserImage
Header
Footer
ImagesUploader
imageList
ErrorMessage
handleImageDelete
handleImageUpload
ImagesUploader
imageList
ErrorMessage
handleImageDelete
handleImageUpload
FileUploadButton
handleFileChange
FileUploadButton
handleFileChange

Login:

The page in which the user can log in with his credentials, if already registered.

The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.

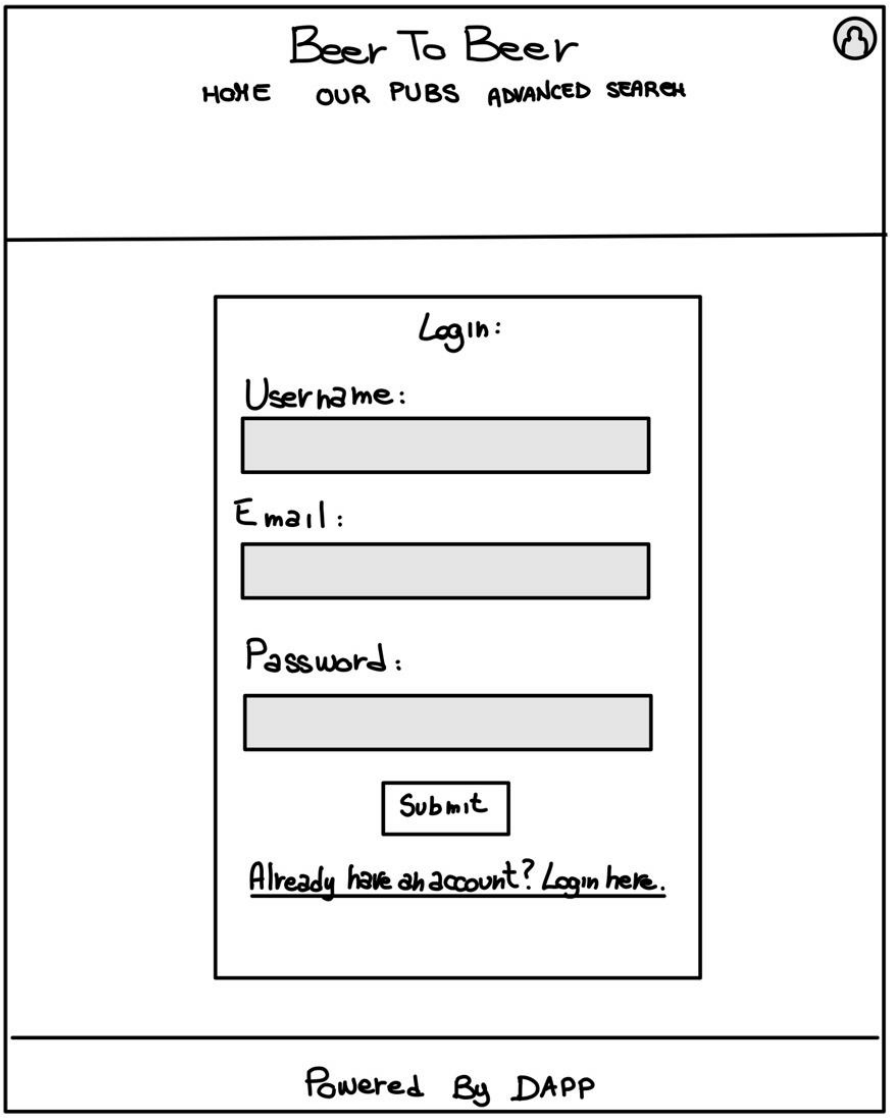


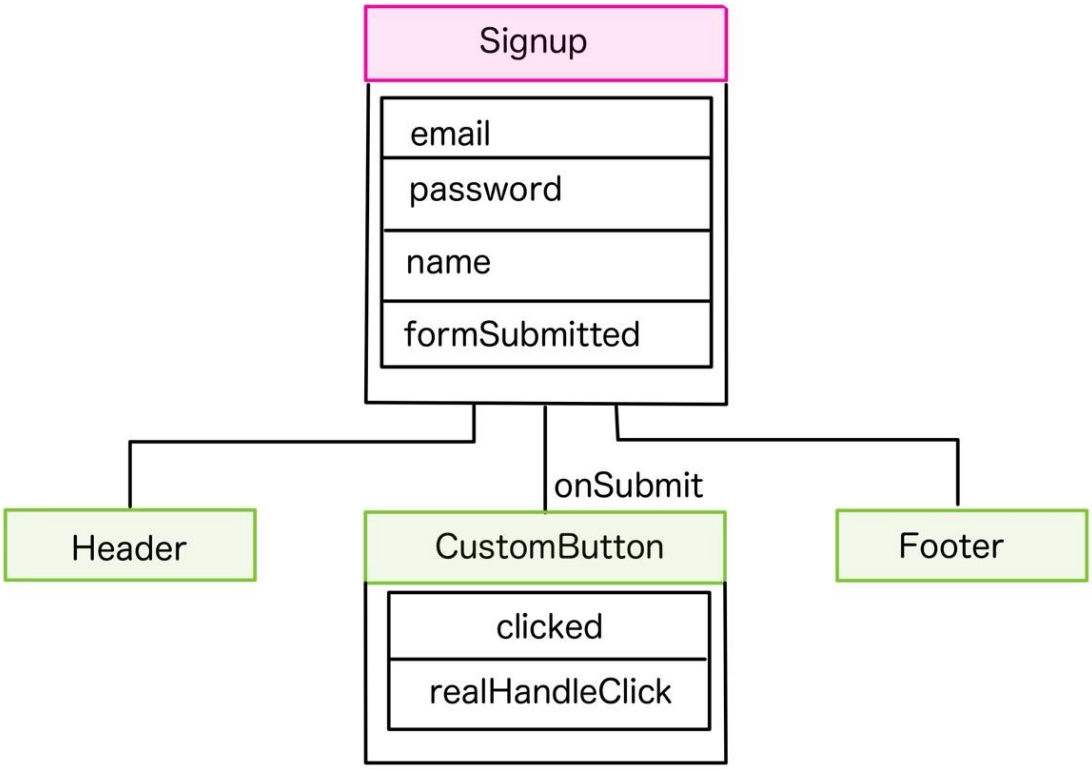

Login
email
password
correctlySubmitted
formSubmitted
onLogin
Header
CustomButton
clicked
realHandleClick
Footer

Signup:

The page in which the user can register himself in the application, so he can have the ability to leave a review of a beer or a pub and write replies to other's ones.

The following pictures represent the mockup and the structure of the page respectively. The mockup provides a visual representation of the page layout, while the structure represents the underlying components: in this representation are displayed each children of a component with the same color; every component has inside it its local states and the handlers it's using; on the black lines there are the handlers it is passing to its children components and the blue ingoing ones represent the access of that component to the Redux Store.




Signup
email
password
name
formSubmitted
onSubmit
Header
CustomButton
clicked
realHandleClick
Footer