### **Лабораторная работа 5 React. Маршрутизация в React - React Router**

**Полезные ссылки**

1. [Обработка событий](https://react.dev/learn/responding-to-events#adding-event-handlers)
2. [Обновление объектов в State](https://react.dev/learn/updating-objects-in-state)
3. [Введение в хуки](https://react.dev/reference/react/hooks)

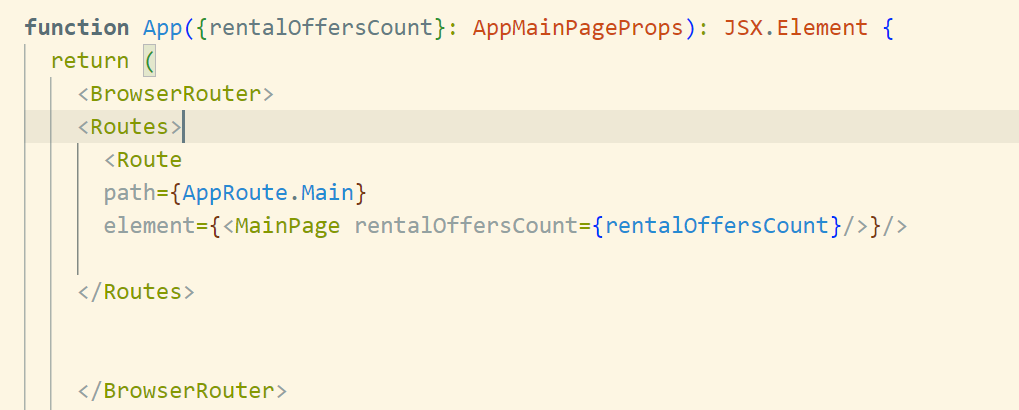
### Задание 1. Добавление маршрутизации

1. Для дальнейшей работы перейдите в папку client. Теперь приступаем к доработке клиентского приложения.
2. Для добавления маршрутизации в приложение необходимо в файл app.tsx импортировать все созданные страницы приложения, такие как: Favorites, Login, Offer, PageNotFound.
3. Также необходимо импортировать компоненты из пакета react-router-dom: BrowserRouter, Route, Routes.
4. С помощью перечисленных ниже компонентов опишите все маршруты в приложении. По каждому адресу должен отрисовываться соответствующий компонент страницы.
   1. Main /
   2. Login /login
   3. Favorites /favorites
   4. Offer /offer/:id

В файле src/const.ts добавим переменную AppRoute, в которой сохраним маршруты



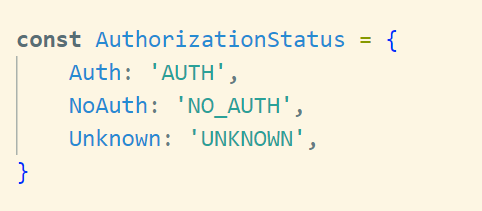
1. В файле app.tsx импортируем переменную AppRoute.
2. Теперь необходимо описать маршрутизацию в приложении и опишем уже имеющийся компонент главной страницы.

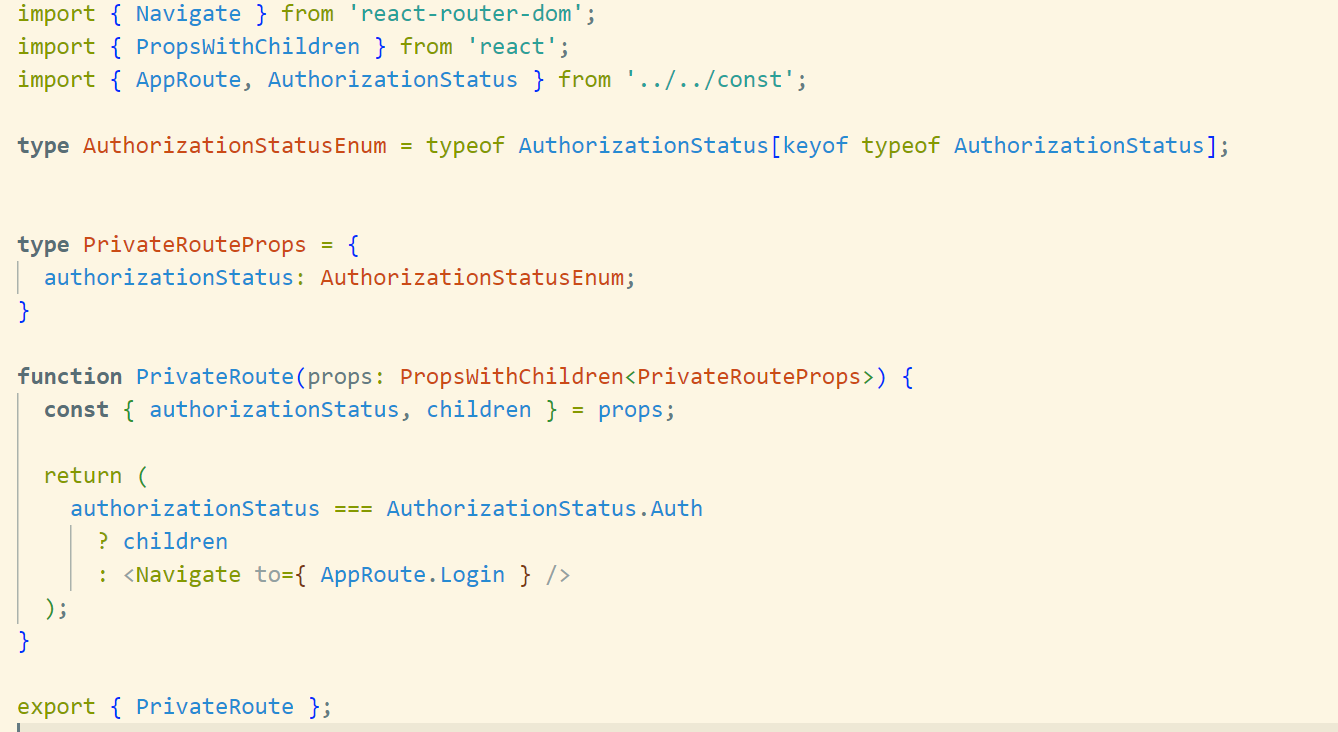


1. Добавьте новые Route для каждой страницы аналогичным образом.
2. Предусмотрите вариант, когда пользователь обращается к несуществующей странице. В данном случае значением свойства path будет \*.

### Задание 2. Создание приватного компонента

При обращении к странице Favorites пользователя, в случае, если он не авторизован, необходимо переадресовывать на страницу Login. Для этой реализации необходимо создать компонент для описания приватных маршрутов. Далее будем использовать этого компонент для реализации обращения к странице Избранное.   
1. Создадим в файле const.ts переменную для статуса авторизации



1. Импортируем созданную переменную в app.tsx
2. Добавим компонент приватного маршрута. В папке components создадим папку private-route, и в ней файл private-route.tsx. Добавим следующий код:   
   

Здесь Navigate - компонент, который перенаправляет пользователя на другой маршрут.

1. Теперь изменим маршрутизация в app.tsx. для страницы Favorites. Для этого импортируем приватный компонент и изменим описание у favorites

<Route

path={ AppRoute.Favorites }

element={

<PrivateRoute

authorizationStatus={ AuthorizationStatus.NoAuth }

>

<Favorites />

</PrivateRoute>

}

/>

### Задание 3. Добавление компонента Логотип - Logo

1. Создайте компонент Logo. Замените в коде теги а на Link  
   
2. Добавьте его вместо кода на все страницы, где он встречается.

### Задание 4. Добавление моковых данных для предложений по аренде

1. В /src создайте новую директорию mocks, а в ней новый файл offers.ts. В нем будут приведены тестовые данные для 4 любых предложений по аренде недвижимости. Эти данные потребуются как для отрисовки карточек с предложениями в списке, так и для отдельных страниц с полной информацией по предложению аренды.   
   Каждая карточка в списке предложений содержит информацию:

**id**: Идентификатор объекта - тип данных string;

**title**: Название объекта - тип данных string;

**type**: Тип жилья Одно из предопределенных значений: apartment (Apartment), room (Private Room), house (House), hotel (Hotel) - тип данных string;

**price**: Стоимость за ночь - number;

**city**: Город - тип данных объект, который представляет собой название города, широту и долготу города;

**location**: тип данных объект, представляющий собой широту и долготу объекта недвижимости

**isFavorite**: Избранное - boolean;

**isPremium**: Премиальность - boolean;

**rating**: Рейтинг - number;

**description**: Описание объекта - string;

**bedrooms**: количество спален - number;

**goods**: список бытовых предметов в квартире - string[ ];

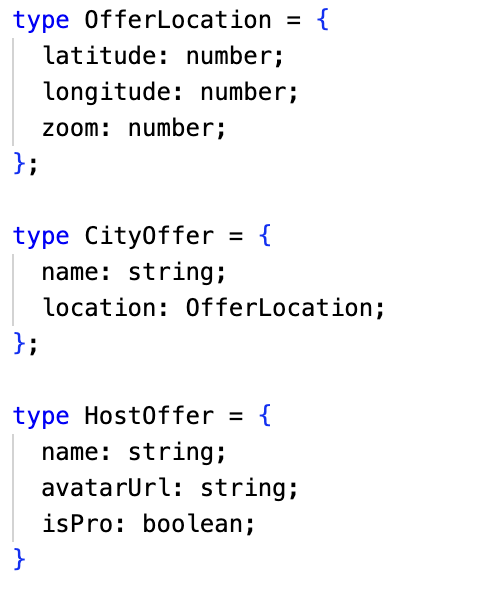
**host**: Информация о хозяине: аватарка, имя, отметка pro (звездочка возле аватарки) и подпись Pro под именем хозяина;

**images**: Фотографии жилья - string[ ];

**maxAdults**: -максимальное количество гостей - number;

В первую очередь создаем типы для описания предложения по аренде. Создадим в папке src папку types, а в ней файл offer.ts.

Так как некоторые типы данных в описании жилья являются объектами, сначала опишем их.



**OfferLocation** - описывает расположение жилья

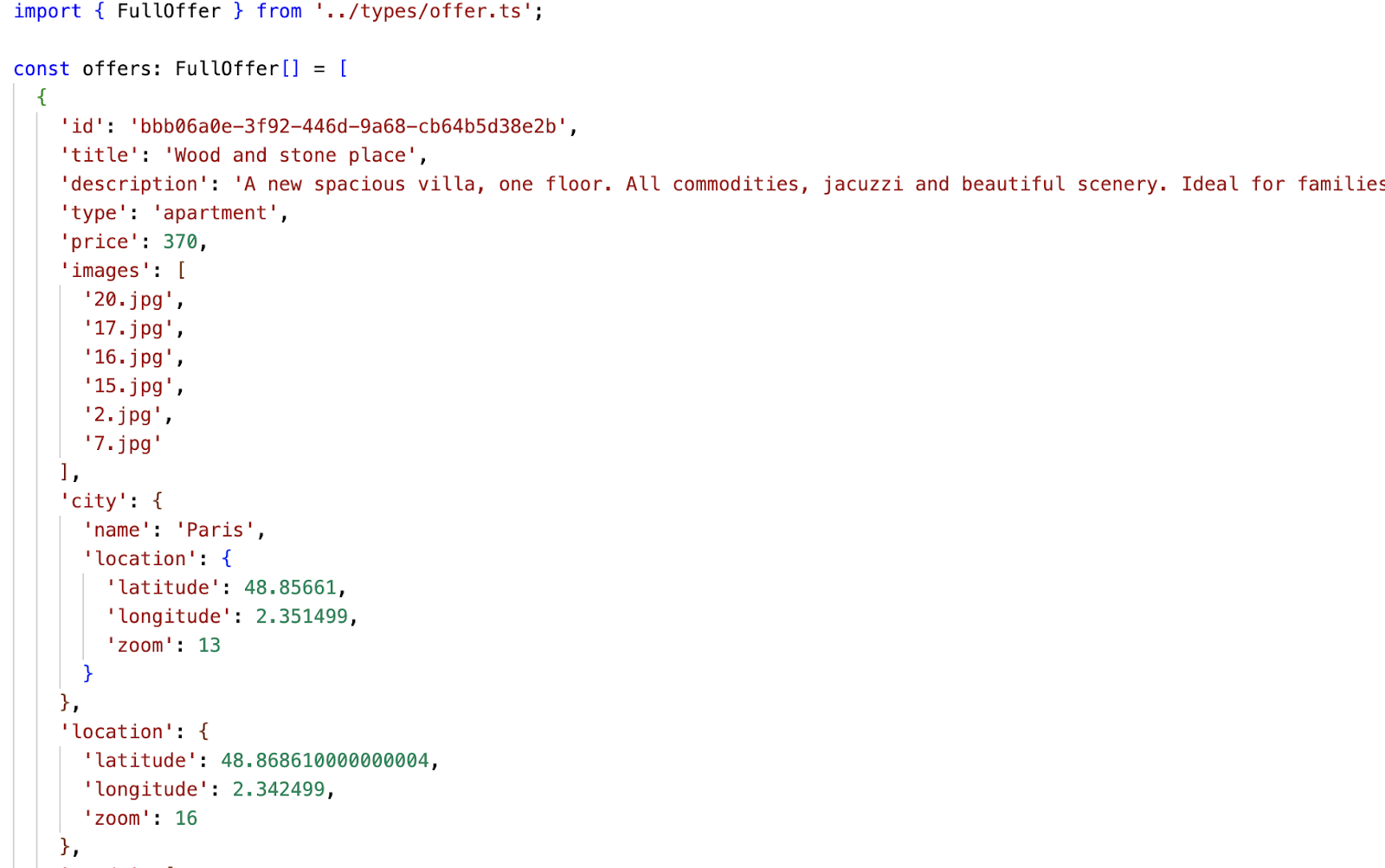
**CityOffer** - описывает город и его координаты

**HostOffer** - описывает хозяина жилья.

Затем создадим тип **FullOffer**, который представляет собой полное описание жилья.



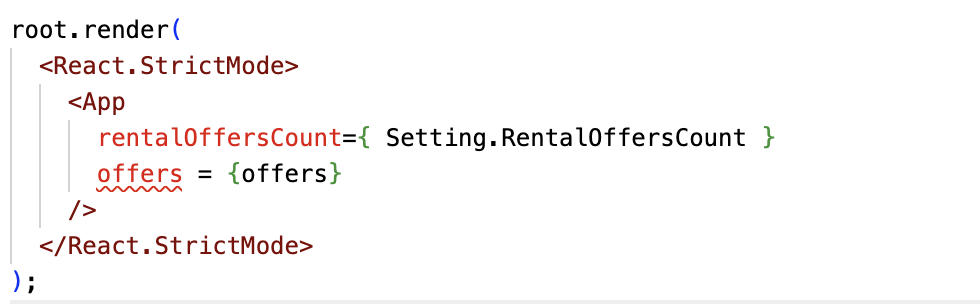
1. Теперь перейдем в mocks/offers.ts. Необходимо описать 4 объекта недвижимости. Импортируем тип FullOffer, а затем создаем переменную, которая представляет собой массив из объектов FullOffer. Фотографии можете подобрать самостоятельно и поместить их в папку public/img. Названия файлов должны отражать суть. Некоторые фото уже есть в папке. Ниже приведен пример 1 предложения жилья. По аналогии создайте еще 3. Не забудьте разнообразить города, в которых расположено жилье (в пределах тех, которые определены на главной странице)





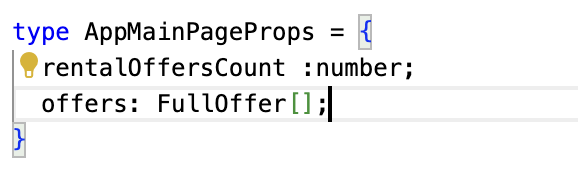
### Задание 5. Подключение моков на страницу

Необходимо подключить созданные моки в index.tsx. Передать их в виде props основному компоненту приложения (App).

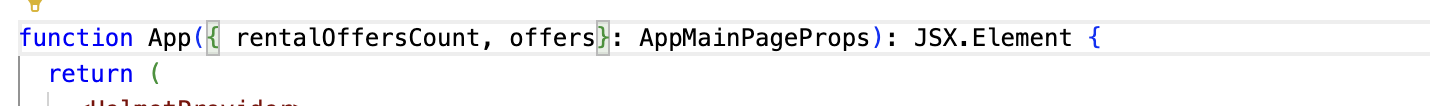
1. Подключим данные в файл index.tsx. Импортируем моковые данные 
2. И передаем через одноименный props в компонент app

Ошибка в строке с props offers будет, так как еще не описан данный props в компоненте App.

1. Перейдем в компонент App. Здесь необходимо импортировать тип FullOffer. Затем обновляем AppMainPagesProps



и передаем в компонент

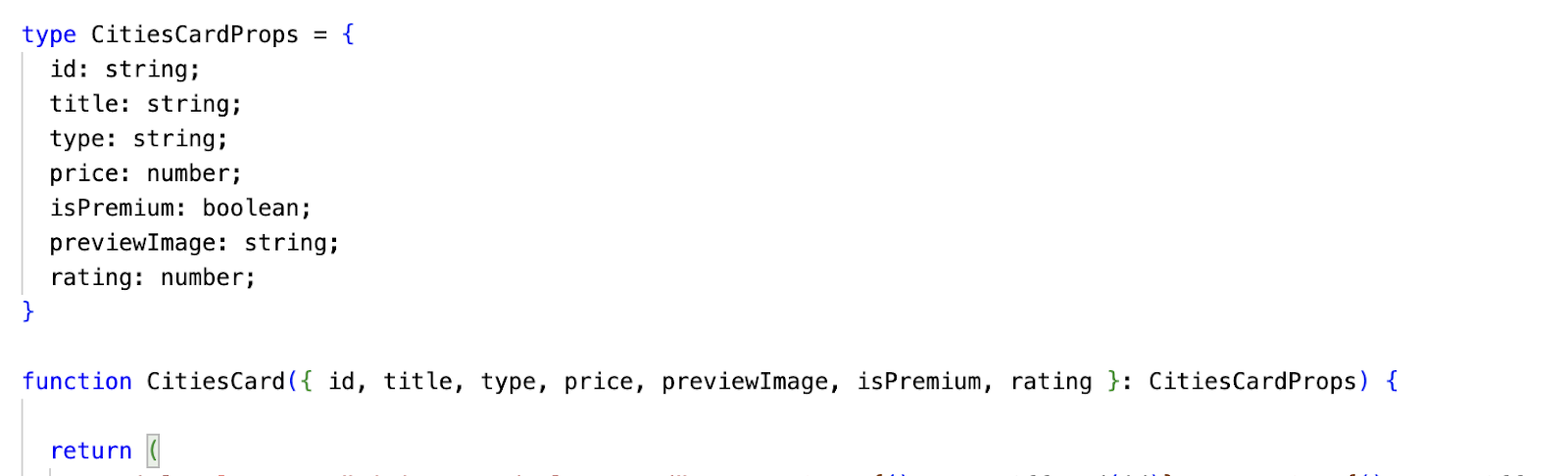


После добавление props offers ошибка в файле index.tsx должна была исчезнуть.

### Задание 6. Обновление компонента «Карточка предложения» (CitiesCard)

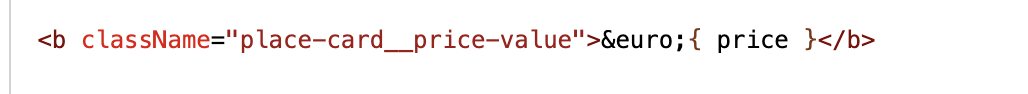
Необходимо добавить возможность получения данных о предложении аренды через props в компоненте CitiesCard.

1. Рассмотрите внимательно компонент карточка предложения. Какие данные необходимы для отрисовки карточки? Каждый элемент массива offers моковых данных представляет собой полное описание предложения. Но на карточке предложения все данные не нужны. Необходимо использовать только *id, название, тип, цена, премиальность, главное фото и рейтинг*. Для этого будем использовать деструктуризацию. Опишем тип props и передадим необходимые данные в карточку предложения.



1. Теперь заменим внутри компонента данные на переданные props.

Например:

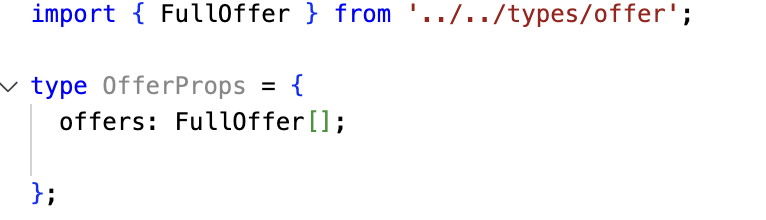


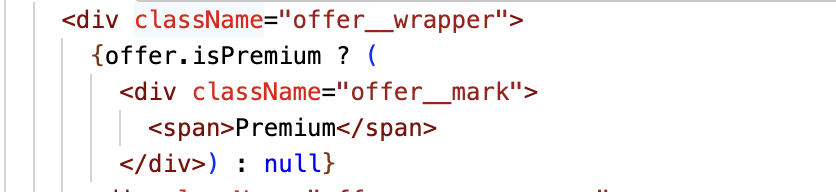
1. Остальные props пропишите самостоятельно.
2. Пока на карточке данные отображаться не будут и на главной странице появятся ошибки, свидетельствующие, что данные в компонент CitiesCard еще не переданы.

### Задание 7. Обновление страницы «Предложение об аренде» (Offer).

Необходимо добавить возможность получения данных о полном предложении аренды через props. Реализуйте по аналогии и предыдущим заданием.

1. Импортируйте тип FullOffer
2. Опишите тип props



1. Передайте props offers на страницу и используйте нотацию, например, offers.price и т.д. где это необходимо (код можно посмотреть в задание 7 п.2)
2. В случаях, где полем объекта является массив (изображения, список бытовых нужд и т.д.) можно использовать функцию [map](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/map)().   
   
3. В тех местах, где используется значение типа boolean, можно использовать следующий способ реализации  
   

### Задание 8. Создание компонента Список предложений

Компонент должен отрисовывать все карточки с предложениями аренды с помощью компонента «Карточка предложения». Все необходимые данные компонент принимает через props.

1. Создайте для компоненты CitiesCardList соответствующую папку и файл
2. В файле types/offer.ts опишите тип для списка предложений



1. Вернитесь в компонент Список предложений и опишите его



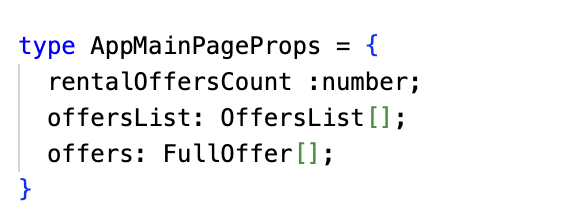
1. Теперь можно обновить код компонента «Главная страница». Замените все компоненты <CitiesCard/> на компонент CitiesCardList.
2. Пришло время передать моковые данные для списка предложений в компонент. Так как каждый элемент списка предложений представляет собой лишь часть от полного предложения по аренде, то создадим новый массив объектов.

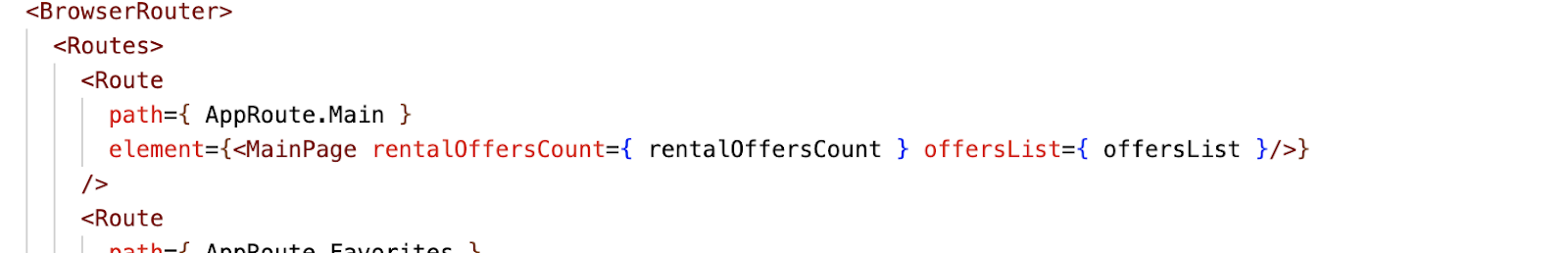
В папке mocks создадим файл offers-list.ts. В нем опишите несколько элементов массива, как в примере.



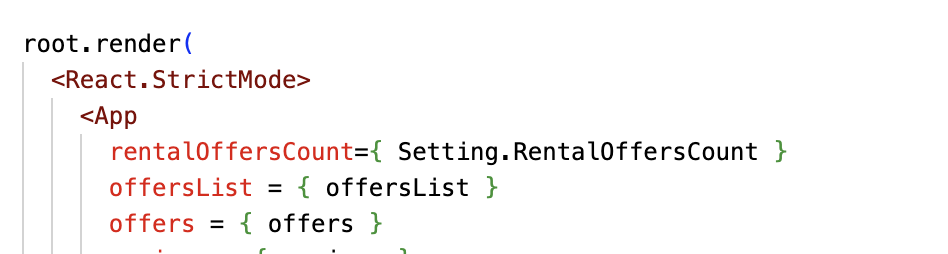
### Задание 9. Передайте моковые предложения по аренде с помощью props компоненту «Главная страница»

1. Добавим в компонент App тип OfferList и передадим props на главную страницу





и опишем также в файле index.tsx



1. На главной странице опишем props offersList



А затем передадим его в компонент CitiesCardList

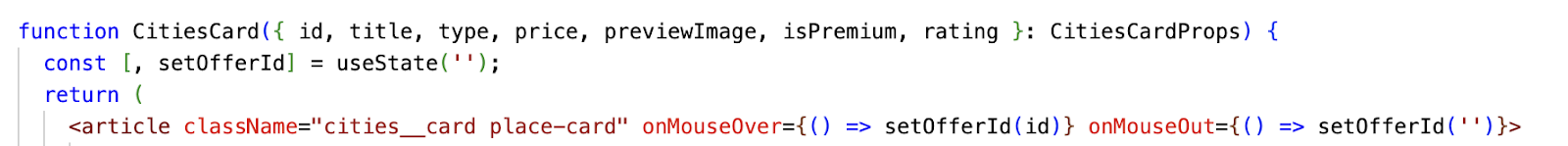


1. Запустите проект. На главной странице должны быть отрисованы предложения по аренде.

### Задание 10. Задание состояния объекта

Добавим в состояние компонента активную карточку с предложением (объект с данными предложения или только уникальный идентификатор предложения). Под активной карточкой предложения подразумевается карточка, на которую пользователь навел курсор.

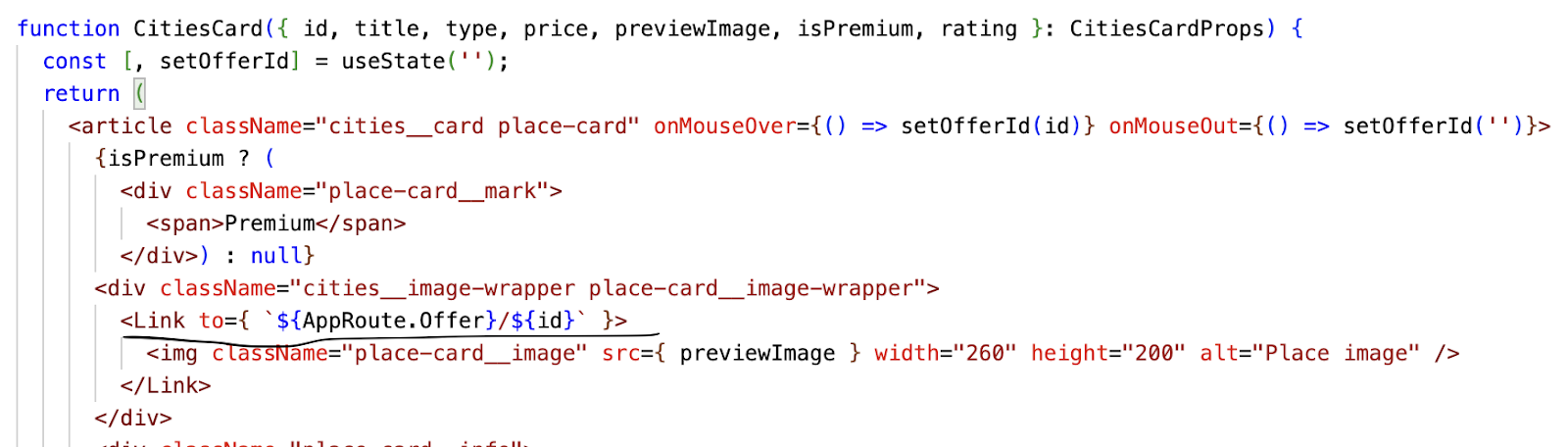
1. Перейдите на компонент CitiesCard и добавьте описание состояния и обработчики событий.



1. Перейдите на страницу Offer. Добавьте хук useParams. Он будет необходим при переходе с главной страницы на выбранное предложение об аренде.

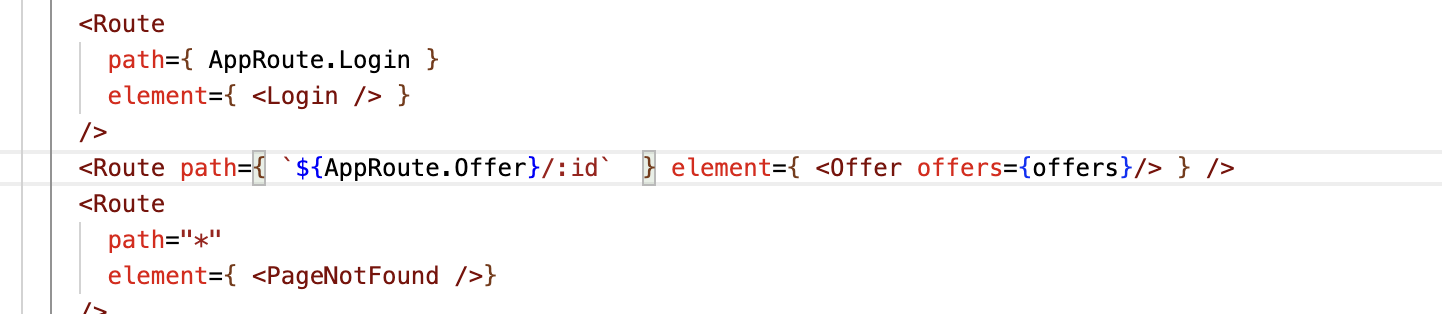


1. Для реализации перехода на нужное предложение об аренде необходимо в компоненте CitiesCard изменить переход по ссылке на самом изображении.



При этом необходимо скорректировать в файлe const.ts путь

1. Добавим новую маршрутизацию и в компонент App



1. Запустите проект и попробуйте перейти с главной страницы на страницу выбранного предложения.

Задание 11. Реализуйте самостоятельно страницу Favorites. Для этого понадобится создать компонент favorite-card и favorite-card-list.

Задание 12. Создайте новый компонент «Форма отправки комментария». Разметка для компонента находится в файле offer.html. Замените на странице «Offer» часть разметки на вновь созданный компонент.