به نام خدا

مستندات کد منبع

EXTRA TIME

پروژه مهندسی نرم‌افزار

زمستان99

محمد حسین شریعتی پور

پوریا ذوالفقاری

سجاد شهرابی

محمد رضا نظری

فهرست

مقدمه

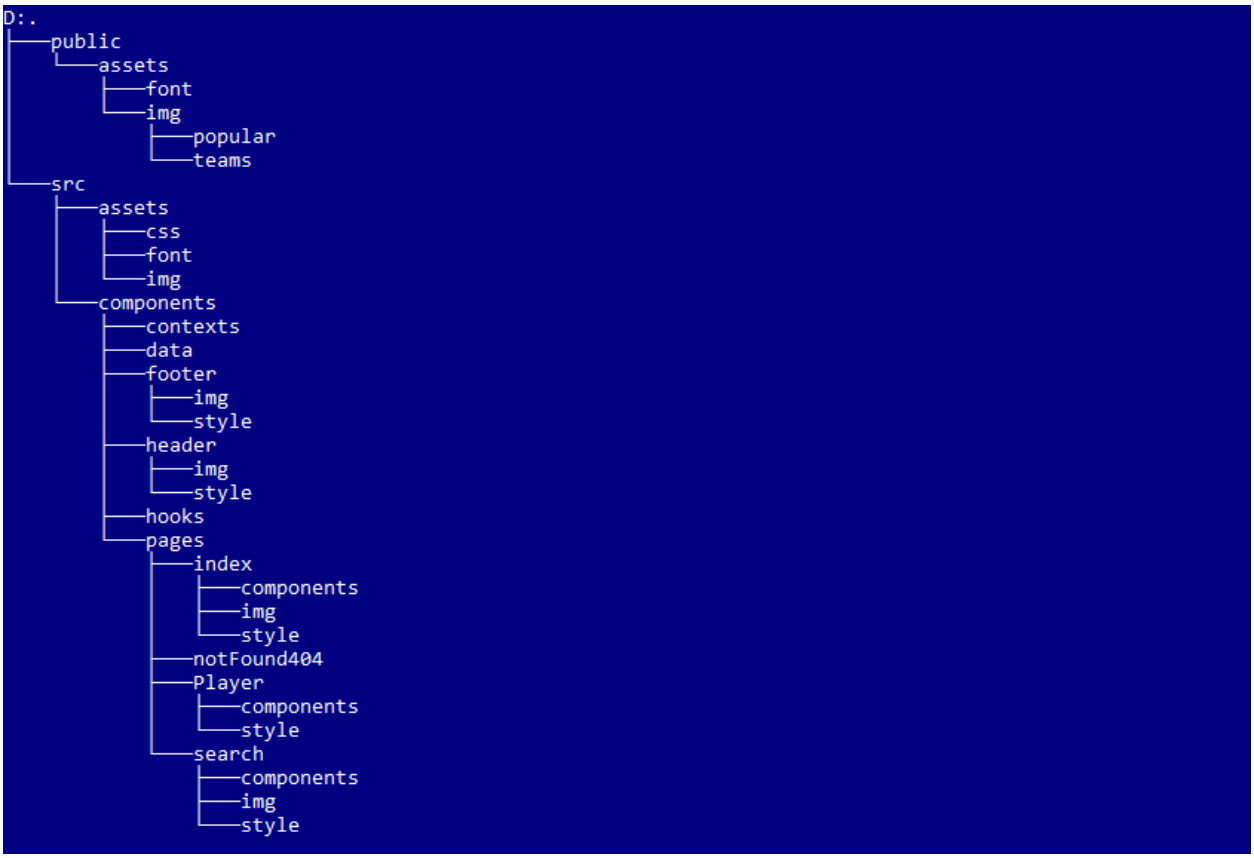
پروژه پیاده سازی شده یک اپلیکیشن تحت وب می‌باشد که به کمک کتابخانه React.js توسعه پیدا کرده است.

دیتاست مورد استفاده در این پروژه شامل اطلاعات بیش از 17 هزار بازیکن بازی FIFA 2021 می‌باشد که به کمک این اپلیکیشن می توان اطلاعات هر یک از این بازیکنان را بصورت بصری مشاهده نمود.

از دیگر خدمات این اپلیکیشن می توان به موارد زیر اشاره نمود:

- جستجو در میان بیش از 17 هزار بازیکن
- افزودن بازیکنان به لیست علاقه‌مندی
- مشاهده برترین بازیکنان در هر پست
- قابلیت مرتب سازی بازیکنان بر حسب قدرت و سن
- و ...

ساختار درختی پروژه



Public: این پوشه شامل فایل های استاتیک مانند فونت‌ها، لوگو و... برنامه می‌باشد.

Src: این پوشه شامل فایل های داینامیک مانند کامپوننت ها، استایل های هر کامپوننت و... می‌باشد.

src/components : محل قرار گیری کلیه کامپوننت‌های نوشته شده می‌باشد، هر کامپوننت یک پوشه به نام خودش دارد.

src/components/contexts: محل قرارگیری stste های سراسری

فایل index.html

در این فایل HTML یک div با id=”root“ وجود دارد. این عنصر، عنصری است که کل برنامه ی ما درون آن قرار می گیرد:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>EXTRA TIME</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>
</html>
```

مواردی همچون تعریف متاتگ ها، عنوان سایت نیز در این فایل نوشته می‌شود.

کامپوننت‌ها

کامپوننت ها قسمت اصلی هر اپلیکیشن توسعه داده شده توسط React js می‌باشند. کلیه کامپوننت های نوشته شده در این پروژه بصورت Functional Components هستند.

هر کامپوننت شامل یک فایل با فرمت js. می‌باشد که شامل موارد زیر می‌باشد:

1. در ابتدای فایل کلیه کتابخانه ها، داده ها، style ها و سایر نیازمندی ها برای پیاده سازی هر کامپوننت import می‌شوند.
2. بدنه کامپوننت که در واقع یک تابع javaScript می‌باشد.
3. Export کردن کامپوننت برای استفاده آن در سایر کامپوننت ها.

سورس کد زیر نمونه ای از کامپوننت نوشته شده می‌باشد:

```
import React from 'react'
import ContainerPage from "./../ContainerPage"
import {Link} from "react-router-dom";

const NotFound = (props)=>{
    return(
        <ContainerPage pageName="player">
            <div className="not-found-player">
                <div className="img">
                    <img src="/soccor.svg" alt="soccor"/>
                </div>
                <h3 className="font-title color-brown">
                    Page Not Found
                </h3>
                <Link to="/" className="btn-home">Go to home</Link>
            </div>
        </ContainerPage>
    )
};

export default NotFound;
```

کامپوننت App.js

خارجی ترین کامپوننت ما می‌باشد که سایر کامپوننت ها در داخل این کامپوننت مورد استفاده قرار می‌گیرند. این کامپوننت در فایل index.js به فایل اصلی برنامه یعنی index.html که در پوشه Public قرار دارد اضافه می‌شود:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import './assets/css/init.css';


ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);


reportWebVitals();
```

Style کامپوننت ها

Style ها بخش مهمی از پیاده سازی هر کامپوننت محسوب می‌شوند که به هر کامپوننت جلوه و ظاهری کاربر پسند می‌دهند. معمولا برای نوشتن style ها از CSS که یک زبان نشانه گذاری محسوب می‌شود استفاده می‌شود.

امروزه فریمورک ها و کتابخانه های مختلفی برای سهلوت استفاده از CSS در پروژه ها توسعه داده شده است. در این پروژه نیز برای سهلوت کار کلیه کدهای مربوط به CSS به صورت SCSS که یک پیش پردازنده برای CSS محسوب می‌شود استفاده شده است. به کمک این پیش پردازنده امکان تعریف مواردی مانند متغیرها، توابع و مواردمختلف دیگری در CSS فراهم می‌شود.کدهای نوشته شده بصورت SCSS را ابتدا به CSS کامپایل کرده و سپس از آن ها در کامپونت ها استفاده خواهیم کرد.

واکنش گرا بودن کامپونت‌ها موضوع مهم دیگری است که امروزه با توجه به گسترش استفاده از موبایل ها و تبلت ها در جامعه باید به آن دقت نمود. در CSS از مدیا کوئری ها بدین منظور استفاده می‌شود.

پروژه شامل دو فایل SCSS مشترک برای استفاده در سایر فایل های SCSS می‌باشد:

1. respansive.scss: شامل کد های مربوط به استفاده از مدیا کوئری ها
2. variables.scss: شامل متغیرهای سراسری مانند رنگ‌ها، تابع های سراسری

```scss
$darkBlue : #255965;
$white : #FFFFFF;
$brown : #746223;
$dark: #707070;
$gold: #FADA5E;

$boxShadowWhite : 5px 12px 20px rgba(36,37,38,.13);
$boxShadowWhite2: 0 1px 5px rgba(0, 0, 0, 0.15);
```

این دو فایل در کلیه فایل های SCSS دیگر import می‌شوند تا مورد استفاده قرار گیرند:

```scss
@import "./../../../assets/css/respansive";
@import "./../../../assets/css/variables";
```

علاوه بر این دو فایل برای کامپونت های مربوط به هر یک از صفحات سایت یک فایل style جدا تعریف می شود همچنین یک فایل style برای استفاده مشترک در کلیه صفحات نوشته شده است که در خارجی ترین کامپونت برنامه یعنی App.js، فراخوانی می شود(init.scss).

dependencyها

dependency های استفاده شده در پروژه در فایل package.json مشخص شده است:

```
{
  "name": "fifa",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^1.2.32",
    "@fortawesome/free-solid-svg-icons": "^5.15.1",
    "@fortawesome/react-fontawesome": "^0.1.12",
    "@testing-library/jest-dom": "^5.11.5",
    "@testing-library/react": "^11.1.2",
    "@testing-library/user-event": "^12.2.2",
    "bootstrap": "^4.5.3",
    "react": "^17.0.1",
    "react-bootstrap": "^1.4.0",
    "react-dom": "^17.0.1",
    "react-id-swiper": "^4.0.0",
    "react-input-range": "^1.3.0",
    "react-loader-spinner": "^3.1.14",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.0",
    "react-select": "^3.1.1",
    "swiper": "^6.3.5",
    "universal-cookie": "^4.0.4",
    "web-vitals": "^0.2.4"
  },
}
```

bootstrap

این کتابخانه شامل مجموعه ای از کلاس های آماده CSS می‌باشد که به منظور سهولت و تسریع در فرآیند توسعه برنامه از آن استفاده شده است.

برای استفاده از کلاس های این کتابخانه کافیست فایل CSS این کتابخانه را در کامپوننت مورد نظر import کنیم سپس نام کلاس های موجود در این کتابخانه را در عناصر مختلف قرار دهیم. با توجه به اینکه میخواهیم از کلاس های این کتابخانه در همه کامپوننت ها استفاده کنیم، این کتابخانه را در خارجی ترین کامپوننت (App.js) فراخوانی می کنیم.

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

swiper

از این کتابخانه جهت پیاده سازی اسلایدرهای واکنش گرا استفاده می‌شود. برای استفاده از این کتابخانه ابتدا کامپوننت های مختلفی که در این کتابخانه وجود دارد را import می‌کنیم:

```
import { Swiper, Navigation, Pagination } from 'swiper/swiper.esm';
```

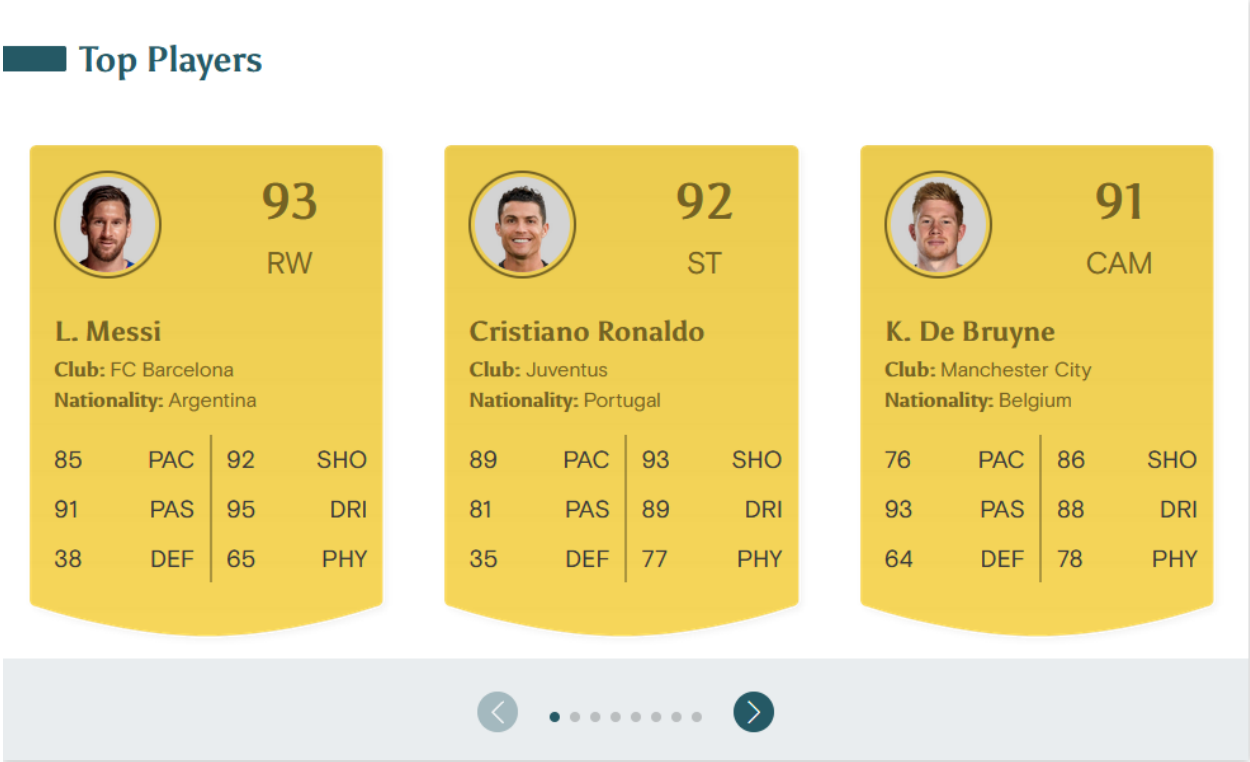
پارامترهای مورد نیاز را تعریف می کنیم، برای مثال به کمک پارامتر breakpoints مشخص می‌کنیم باتوجه به عرض های مختلف صفحه چند اسلاید نمایش داده شود، پارامتر autoplay مشخص می‌کند که اسلایدر بصورت خودکار به اسلاید های بعدی برود.

```
const params = {
        // Provide Swiper class as props
        Swiper,
        // Add modules you need
        modules: [Navigation, Pagination],
        pagination: {
            el: '.swiper-pagination',
            type: 'bullets',
            clickable: true,
        },
        autoplay: {
            delay: 2500,
            disableOnInteraction: false
        },
        navigation: {
            nextEl: '.swiper-button-next.btn-slider',
            prevEl: '.swiper-button-prev.btn-slider'
        },
        renderPrevButton: () => <button className="swiper-button-prev btn-slider"></button>,
        renderNextButton: () => <button className="swiper-button-next btn-slider"></button>,
        spaceBetween: 30,
        breakpoints: {
            // when window width is >= 320px
            320: {
                slidesPerView: 1,
                spaceBetween: 20
            },
            577: {
                slidesPerView: 1,

            },
            // when window width is >= 480px
            768: {
                slidesPerView: 2,
                spaceBetween: 50
            },
            // when window width is >= 640px
            992: {
                slidesPerView: 2,
            },
            1200: {
                slidesPerView: 3,
            }
        },
    };
```

در آخر لازم است این پارامترهای تعریف شده را به کامپوننت ReactIdSwiperCustom بدهیم:

```
<ReactIdSwiperCustom {...params}>
   ...
</ReactIdSwiperCustom>
```

نمونه ای از اسلایدرهای پیاده سازی شده به کمک این کتابخانه در پروژه در تصویر زیر آمده است، این اسلایدر نشان دهنده 10 بازیکن برتر از نظر قدرت می‌باشد:



Fontawesome

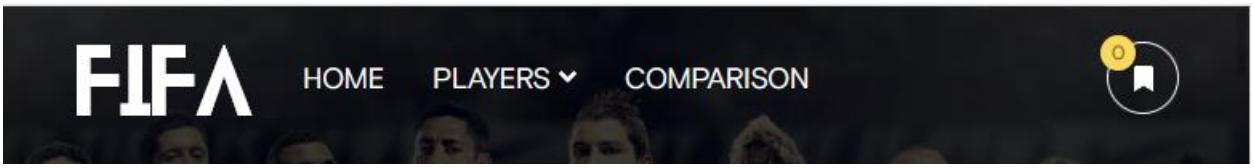
این کتابخانه شامل مجموعه ای از آیکون ها در دسته بندی های مخلف می‌باشد، برای استفاده از آیکون های موجود کامپوننت Fontawesome و کامپوننت مربوط به آیکون مورد نظر را در کامپوننت مقصد import می‌کنیم، برای مثال برای استفاده از آیکون BookMark:

```
import {FontAwesomeIcon} from "@fortawesome/react-fontawesome";
import {faBookmark} from '@fortawesome/free-solid-svg-icons'
```

سپس در بدنه کامپوننت مقصد بصورت زیر از آن استفاده می‌کنیم:

```
<FontAwesomeIcon  icon={faBookmark} />
```

از ایکون BookMark در منو برنامه استفاده شده است:



universal-cookie

همانطور که گفته شد یکی از امکانات برنامه افزودن بازیکنان به لیست علاقه‌مندی می‌باشد. با افزودن هر بازیکن به لیست علاقه مندی اطلاعات این بازیکن شامل:

- Id بازیکن
- نام بازیکن
- آدرس عکس بازیکن

در کوکی کاربر ذخیره شده تا کاربر در مراجعات بعدی بازیکن را در لیست خود داشته باشد.

برای ذخیره کردن کوکی ها در برنامه از پکیج universal-cookie استفاده شده است. برای استفاده از توابع این پکیج ابتدا این پکیج را import می‌کنیم:

```
import Cookies from 'universal-cookie';
```

سپس یه شی از این کلاس می‌سازیم:

```
const cookies = new Cookies();
```

حال برای ذخیره کردن کوکی تابع set و برای دسترسی به مقادیر کوکی تابع get را بر روی این شی فراخوانی می‌کنیم. تابع set شامل سه آرگومان ورودی می‌شود که به ترتیب:

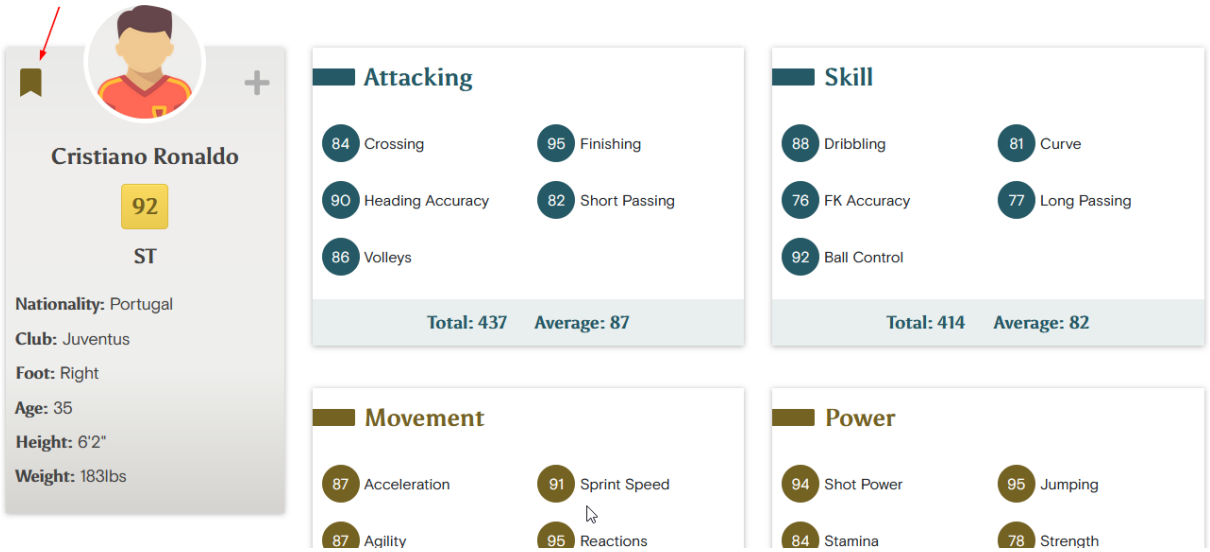
- نام کوکی
- مقادیر کوکی
- مسیر ذخیره کردن کوکی

و تابع get هم شامل یک ورودی که نام کوکی است می‌باشد.

تصویر زیر نمونه ای از کد های استفاده شده برای ذخیره و دسترسی به مقادیر کوکی در برنامه می‌باشد:

```
//Get
const lastPlayer = cookies.get('player') ? cookies.get('player') : [];

//Set
cookies.set('player', temp, { path: '/' });
```

برای افزودن بازیکن به لیست علاقه مندی ابتدا کاربر باید به صفحه معرفی همان بازیکن مراجعه کند، و سپس به کمک آیکون BooKmark قرار داده شده در این صفحه بازیکن مد نظر خود را به لیست اضافه کند:



با کلیک کاربر بر روی این آیکون تابع زیر فراخوانی می‌شود:

```
const togglePlayerBookMark = (id, name, src)=>{
        if(!bookMarkActive){
            let temp = playersBookMark ? playersBookMark : [];
            temp.push({
                id: id,
                name: name,
                src: src
            });
            setPlayerBookMark(temp);
            counterPlus();
            cookies.set('player', temp, { path: '/' });
        }else{
            var temp = playersBookMark;
            for(var i = 0 ; i < temp.length ; i++){
                if (id === temp[i].id) {
                    temp.splice(i, 1);
                }
            }
            setPlayerBookMark(temp);
            counterMines();
            cookies.set('player', temp, { path: '/' });
        }

        setBookMarkActive(!bookMarkActive);
    };
```

سه وروی id، نام و آدرس عکس بازیکن به این تابع داده می‌شود، سپس چک می‌شود، اگر این بازیکن از قبل در لیست وجود داشته باشد از لیست حذف می‌شود (قسمت else اجرا می‌شود) و اگر در لیست نباشد به لیست اضافه می‌شود، در هر دو حالت کوکی مربوطه بروزرسانی خواهد شد.

صفحات برنامه

برای صفحه بندی کردن برنامه از پکیج react-router-dom استفاده شده است. لیست صفحات و کامپوننت مربوط به هر صفحه در فایل Route.js نوشته شده است، برنامه شامل چهار صفحه می‌باشد:

- صفحه اصلی (Index)
- معرفی بازیکن (Player)
- جستجو و فیلتر هوشمند بازیکن ها (Search)
- صفحه 404

```
import Index from "./components/pages/index/index"
import Player from "./components/pages/Player/Player"
import Search from "./components/pages/search/Search"
import NotFound from "./components/pages/notFound404/NotFound"

export var routes = [

    {
        exact: true,
        name : "Home",
        path : "/",
        component : Index,
    },

    {
        name : "Player",
        path : "/player/:playerId",
        component : Player,
    },

    {
        name : "Search",
        path : "/search",
        component : Search,
    },

    {
        name : "404",
        component : NotFound,
    },


];
```

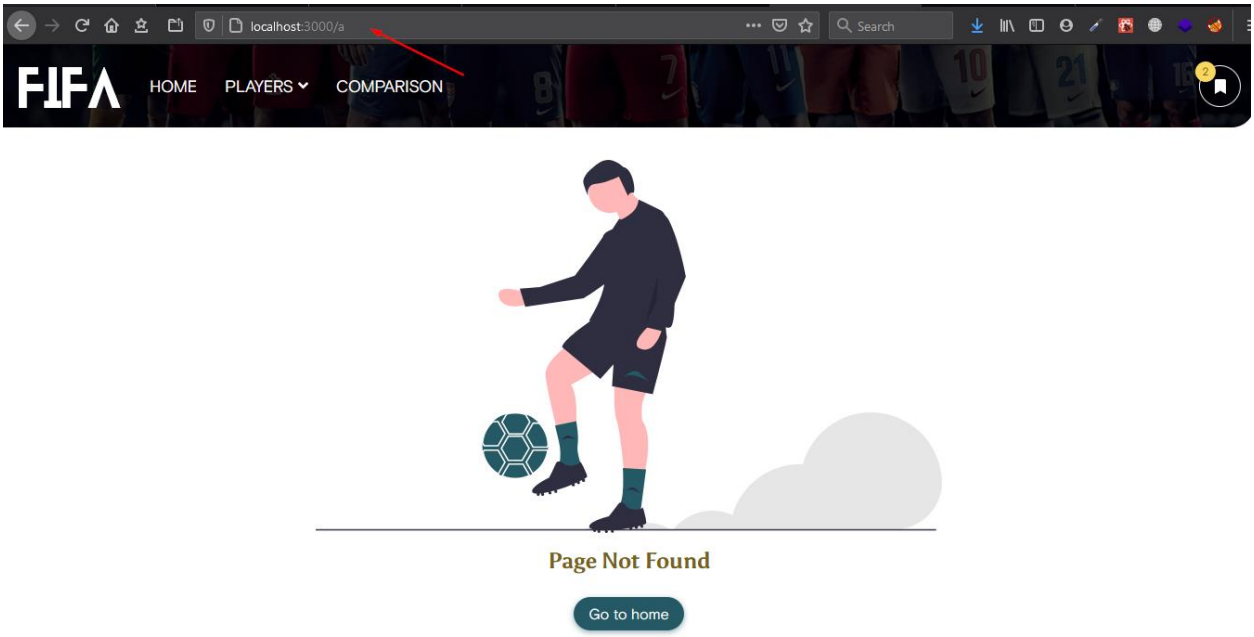
اطلاعات صفحات در آرایه routs ذخیره شده است، این آرایه در کامپوننت App.js، import شده است و با فراخوانی تابع map بر روی این آرایه، اطلاعات هر صفحه به کامپوننت Route داده شده است. کامپوننت Route یکی از کامپوننت های پکیج react-router-dom می‌باشد.

```
import {routes} from "./Route"
import {
    BrowserRouter as Router,
    Switch,
    Route,
} from "react-router-dom";


function App(){


  return (
    <LoaderProvider>
      <BookMarkProvider>
          <CounterPlayersBookMark>
              <Router>
                  <Switch>
                      {
                          routes.map((item,index) =>(
                              <Route key={index} {...item} />
                          ))
                      }
                  </Switch>
              </Router>
          </CounterPlayersBookMark>
      </BookMarkProvider>
    </LoaderProvider>
  );
}

export default App;
```

برای لینک دهی به صفحات از کامپوننت Link که مربوط به پکیج react-router-dom می‌باشد استفاده می‌شود، آدرس صفحه مدنظر در ویژگی to این کامپوننت قرار داده می‌شود، برای مثال برای لینک دادن به صفحه اصلی بصورت زیر از کامپوننت Link استفاده می‌کنیم:

```
<Link to="/" className="logo">
    <img src={Logo} alt="LOGO"/>
</Link>
```

همچنین در صورت وارد کردن آدرس نامعتبر در برنامه، کاربر به صفحه 404 ریدایرکت می‌شود.



استفاده از APIها

برای استفاده از اطلاعات دیتاست، APIهای مختلفی به زبان PHP نوشته شده است و بر روی هاست قرار داده دشه است. برای ارسال درخواست به API ها در کامپوننت‌ها از fetch استفاده شده است، برای مثال برای درخواست اطلاعات 5 دروازبان برتر بصورت زیر عمل شده است:

```
fetch("http://nabzsalamati.ir/FIFA/?Best=GK&num=5")
    .then(res => res.json())
    .then(
        (result) => {
            setPlayerGK(result);
        },

        (error) => {

        }
 );
```