

명함 제조기 클론코딩

김영일

by엘리 2020. 10. 24 - 2020. 11. 15

명함제조기 기획은?

다양한 툴과 라이브러리를 사용해보자

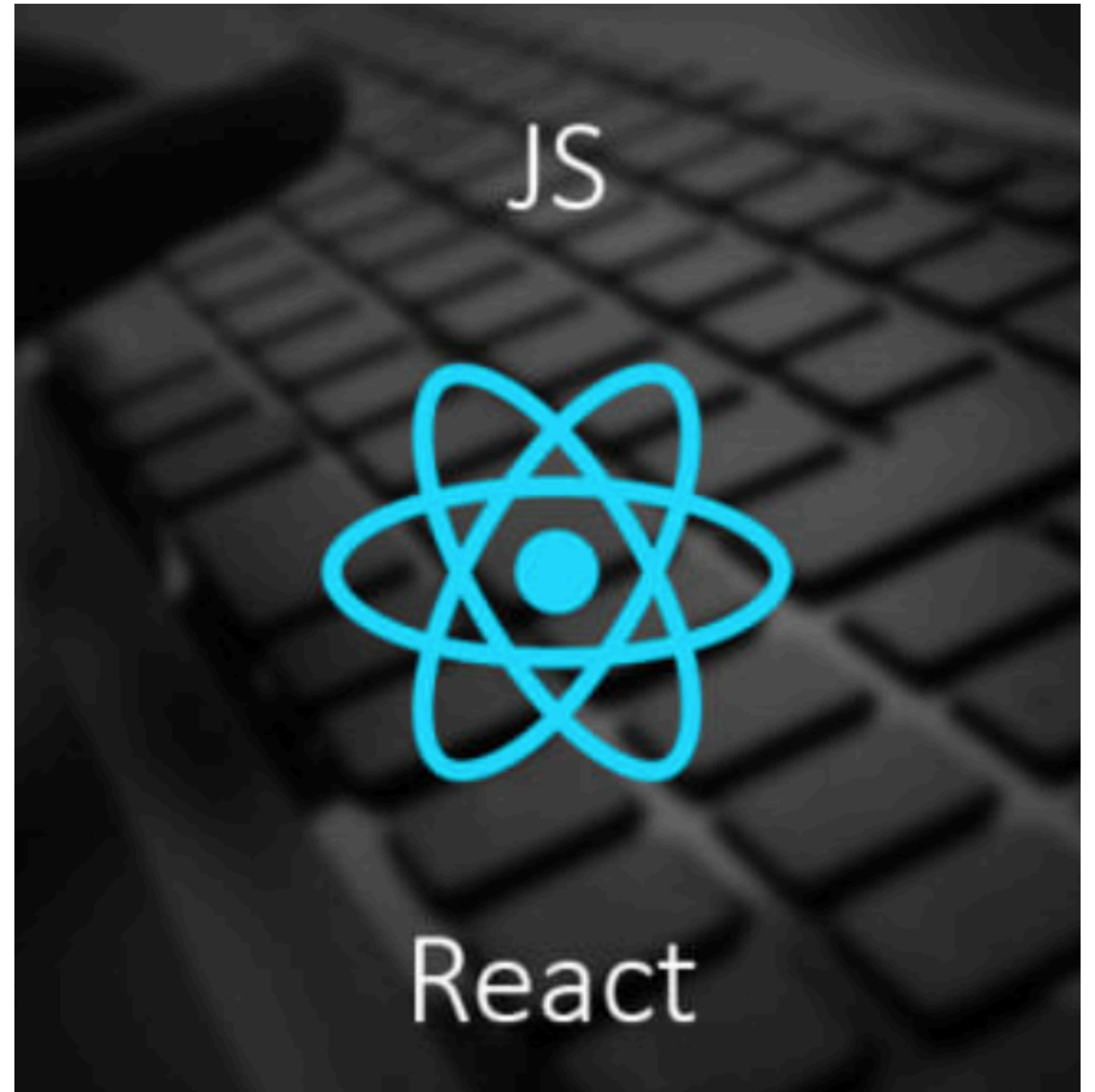
- 파이어베이스 SDK를 사용하여 Auth와 Real time database를 활용.
- 클라우디너리 데이터 베이스를 사용하여 용량이 다소 큰 이미지를 저장.
- ReactJS를 활용해 컴포넌트 단위로 관리.
- PostCSS를 활용해 CSS 모듈화 및 글로벌 셋업.

프론트엔드

React

주요 컨셉

- 리엑트는 컴포넌트들의 집합.
- HTML은 기본적으로 DOM(documents object model) 트리구조를 형성하는데 리엑트는 VDOM(Virtual Documents object model) 을 형성.
- components는 state 가 변경이되면 자동적으로 re-rendering 을 하기때문에 편리함. 그러나 root components 가 변경이 되면 자식 components 들 도 re-rendering 되기때문에 성능저하에 우려가 있음.
- 이걸 방지하기위해 가상의 돔구조(VDOM)를 메모리에 저장하고 변경된 부분만 비교하여 단일적으로 re-rendering을 함. 성능 저하를 극복하고 초당 60프레임 이상 나오게끔 설계가 되어있음.



React

컴포넌트 관리

- 디렉토리 하나당 컴포넌트.jsx파일 1개와 module.css 파일 1개를 쌍으로 관리.
- React-route-dom 라이브러리로 페이지 라우팅을 정의
- 백엔드 데이터는 service 파일안에 따로 클래스형태로 만듦.



The screenshot shows a VS Code editor with the Explorer sidebar on the left and the Editor view on the right. The Explorer sidebar shows a project structure with a 'src' directory containing a 'common' directory and a 'components' directory. The 'components' directory is expanded, showing subdirectories like 'button', 'card', 'card_add_form', 'card_edit_form', 'editor', 'footer', 'header', 'image_file_input', 'login', 'maker', 'preview', and 'service'. The 'service' directory is selected. The Editor view shows the 'app.jsx' file, which contains the following code:

```
1 import React from 'react';
2 import { BrowserRouter, Route, Switch } from 'react-router-dom';
3 import styles from './app.module.css';
4 import Login from './components/login/login';
5 import Maker from './components/maker/maker';
6
7 function App({ FileInput, authService, cardRepository }) {
8   return (
9     <div className={styles.app}>
10       <BrowserRouter>
11         <Switch>
12           <Route exact path="/">
13             <Login authService={authService} />
14           </Route>
15           <Route path="/maker">
16             <Maker
17               FileInput={FileInput}
18               authService={authService}
19               cardRepository={cardRepository}
20             />
21           </Route>
22         </Switch>
23       </BrowserRouter>
24     </div>
25   );
26 }
27
28 export default App;
```

PostCSS

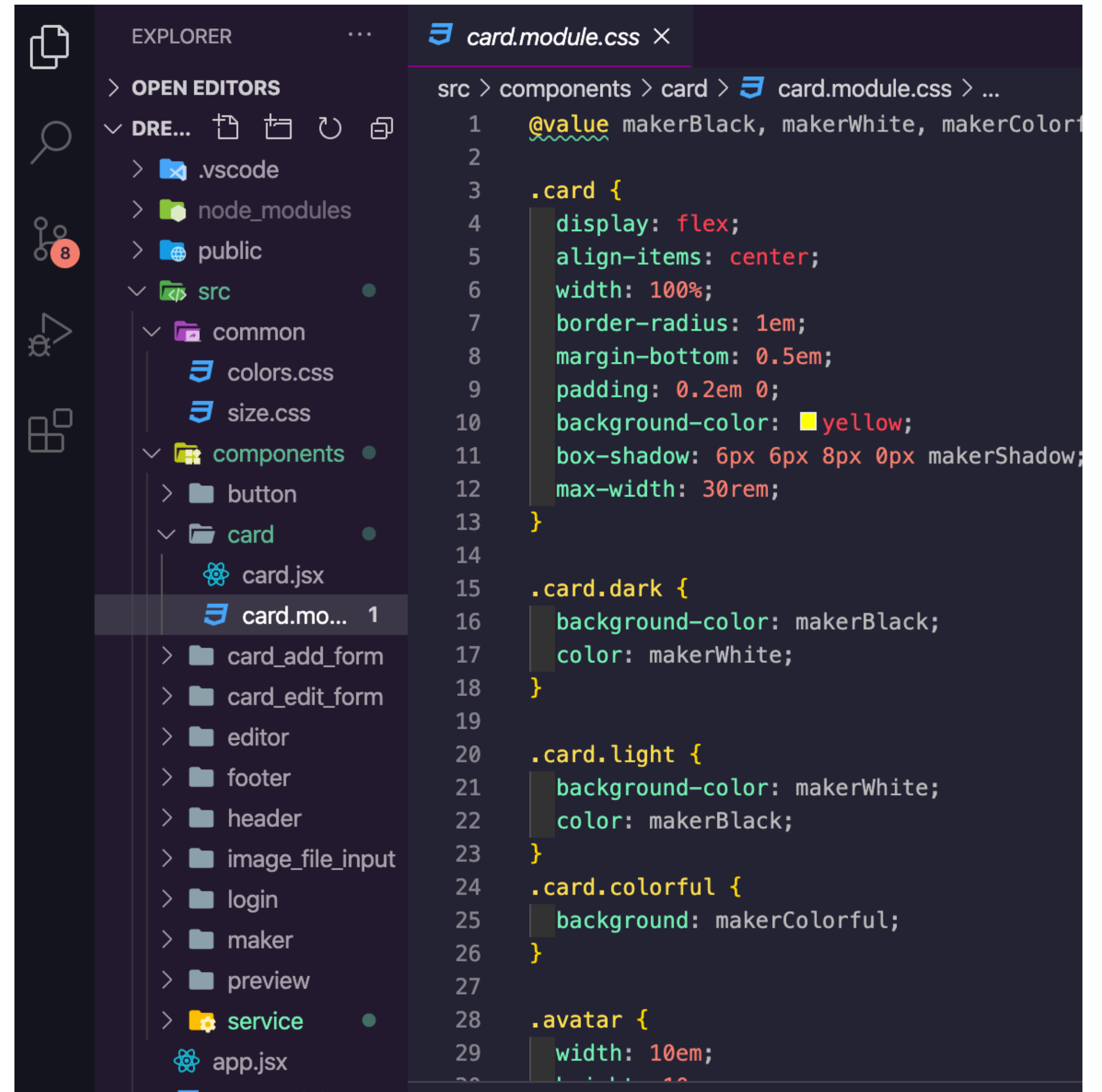
주요 컨셉

- CSS파일을 모듈화 시켜 코드의 가독성 향상
- 컴포넌트 단위로 관리하게되는 리액트와 궁합이 잘 맞음.



PostCSS

- Common 파일을 만들어 global setup을 설정하였음.
- 하나의 컴포넌트와 연결하여 1:1로 매칭하여 사용.
- 상당히 직관적이고 클래스네임으로 고민할 필요가 많이 줄어듦.
- 협업에서 빛을 발휘함.



백엔드

Firebase

주요 기능

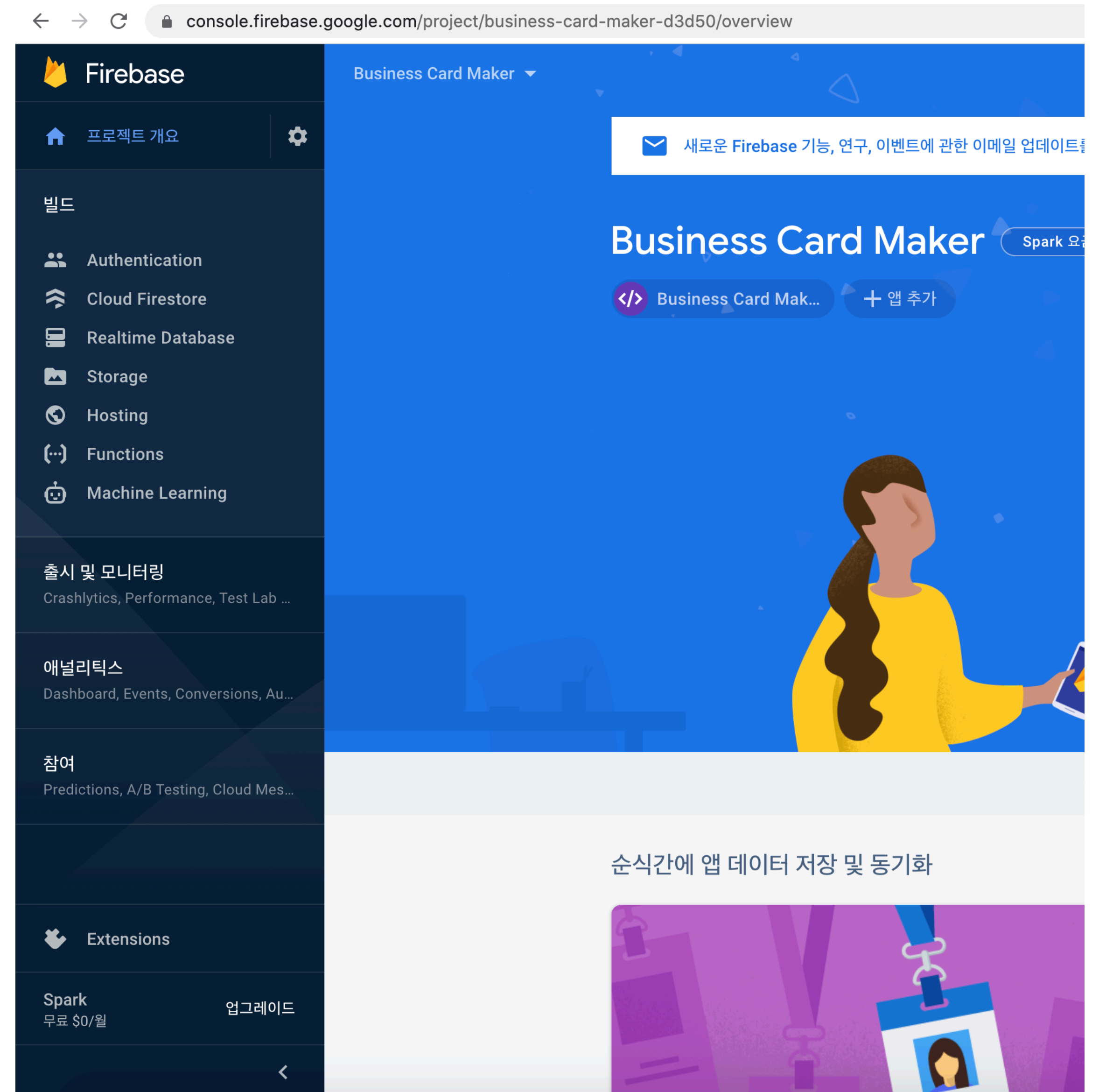
- 인증서비스
- 실시간 데이터베이스
- 스토리지
- 원격 구성
- 푸시알림



Firebase

Firebase console

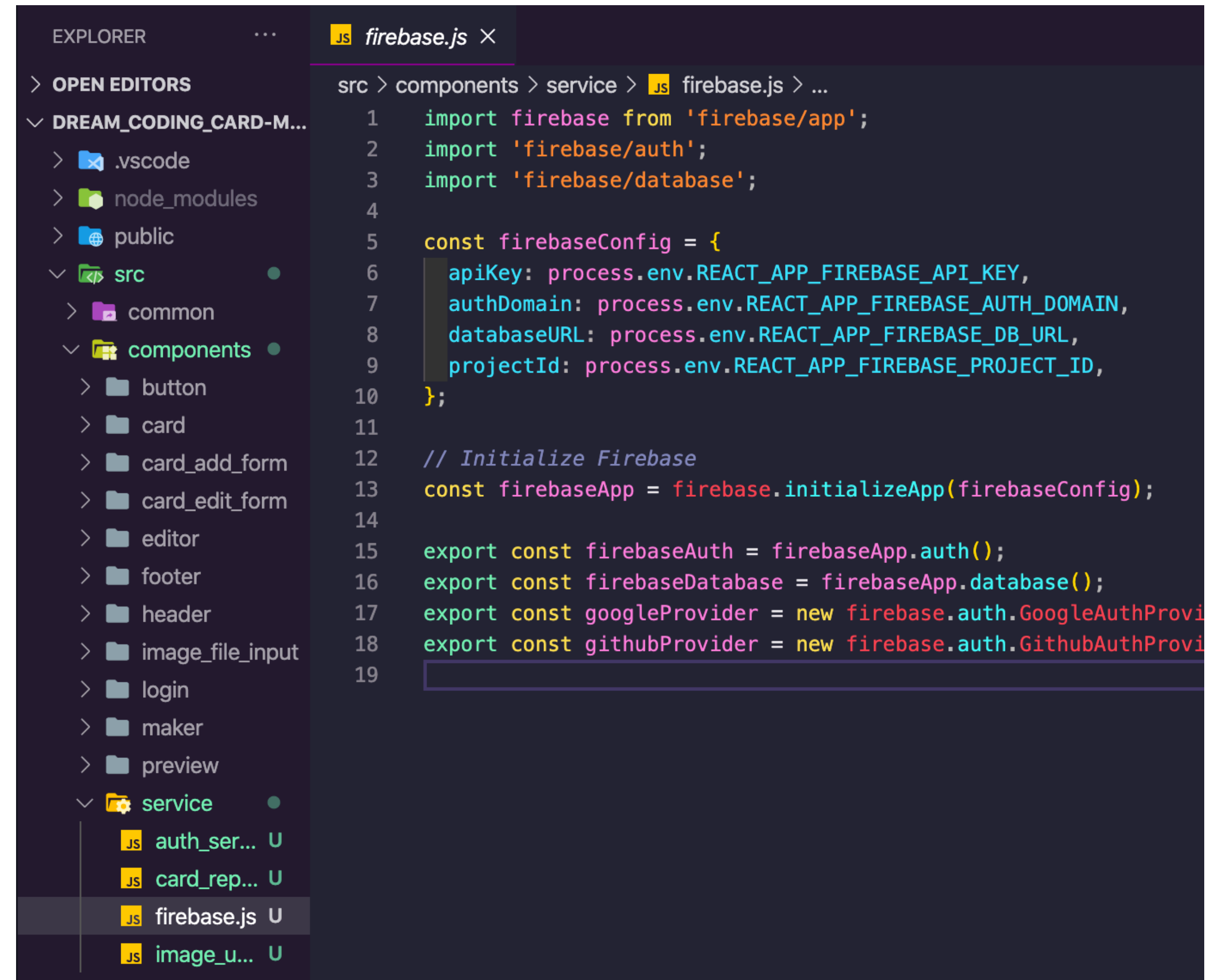
- 편리한만큼 굉장히 무겁기때문에 의존성에 대해서 한번 생각을 해볼 필요도 있음.
- 보안성을 생각했을때는 구글Auth가 내가 관리하는것보다 훨씬 강력할거라고 생각하기 때문에 자주 사용할거같음.
- 가벼운 사이드프로젝트를 수행하기 편하게 만들어주지만 너무 편하기에 개발자로서 좀 멀리하고 싶어짐.



Firebase

디렉토리 구조

- Service 디렉토리 안에서 관리.
- 클래스의 형태로 함수를 작성하여 기능을 구현하고 있음.
- .env를 통한 private key 보안화.



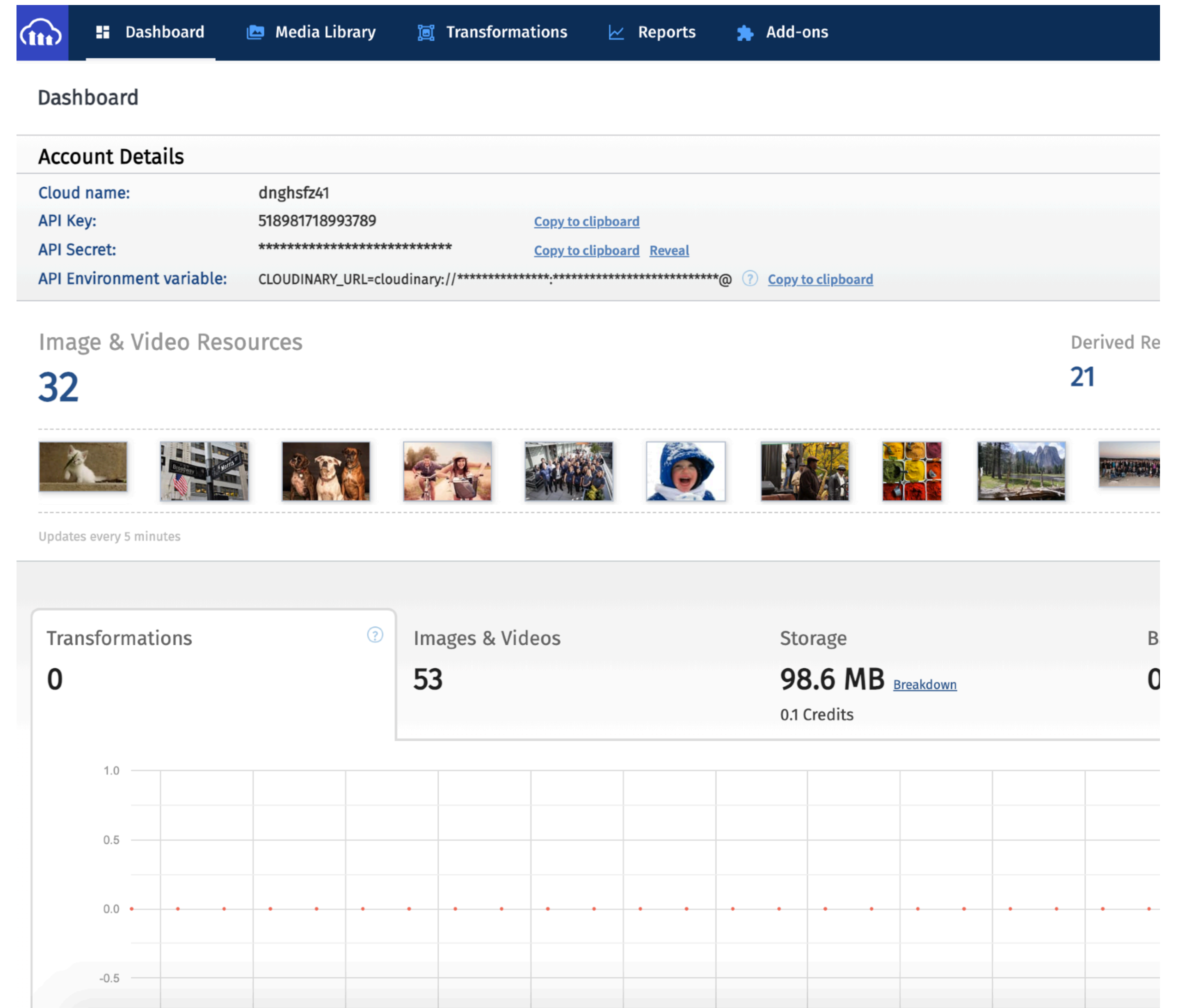
The screenshot shows the VS Code interface. On the left, the Explorer panel displays the project structure. The 'src' directory is expanded, showing subdirectories like 'common', 'components', and 'service'. The 'service' directory is selected, and its contents are listed: 'auth_ser...', 'card_rep...', 'firebase.js', and 'image_u...'. The 'firebase.js' file is highlighted. On the right, the Editor panel shows the content of 'firebase.js'. The code imports 'firebase/app', 'firebase/auth', and 'firebase/database'. It defines a 'firebaseConfig' object with keys for 'apiKey', 'authDomain', 'databaseURL', and 'projectId', all using environment variables from 'process.env'. It then initializes Firebase with 'firebase.initializeApp(firebaseConfig)'. Finally, it exports 'firebaseAuth', 'firebaseDatabase', 'googleProvider', and 'githubProvider' as constants.

```
src > components > service > JS firebase.js > ...
1  import firebase from 'firebase/app';
2  import 'firebase/auth';
3  import 'firebase/database';
4
5  const firebaseConfig = {
6    apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
7    authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
8    databaseURL: process.env.REACT_APP_FIREBASE_DB_URL,
9    projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
10 };
11
12 // Initialize Firebase
13 const firebaseApp = firebase.initializeApp(firebaseConfig);
14
15 export const firebaseAuth = firebaseApp.auth();
16 export const firebaseDatabase = firebaseApp.database();
17 export const googleProvider = new firebase.auth.GoogleAuthProvider();
18 export const githubProvider = new firebase.auth.GithubAuthProvider();
19
```

Cloudinary

Image database

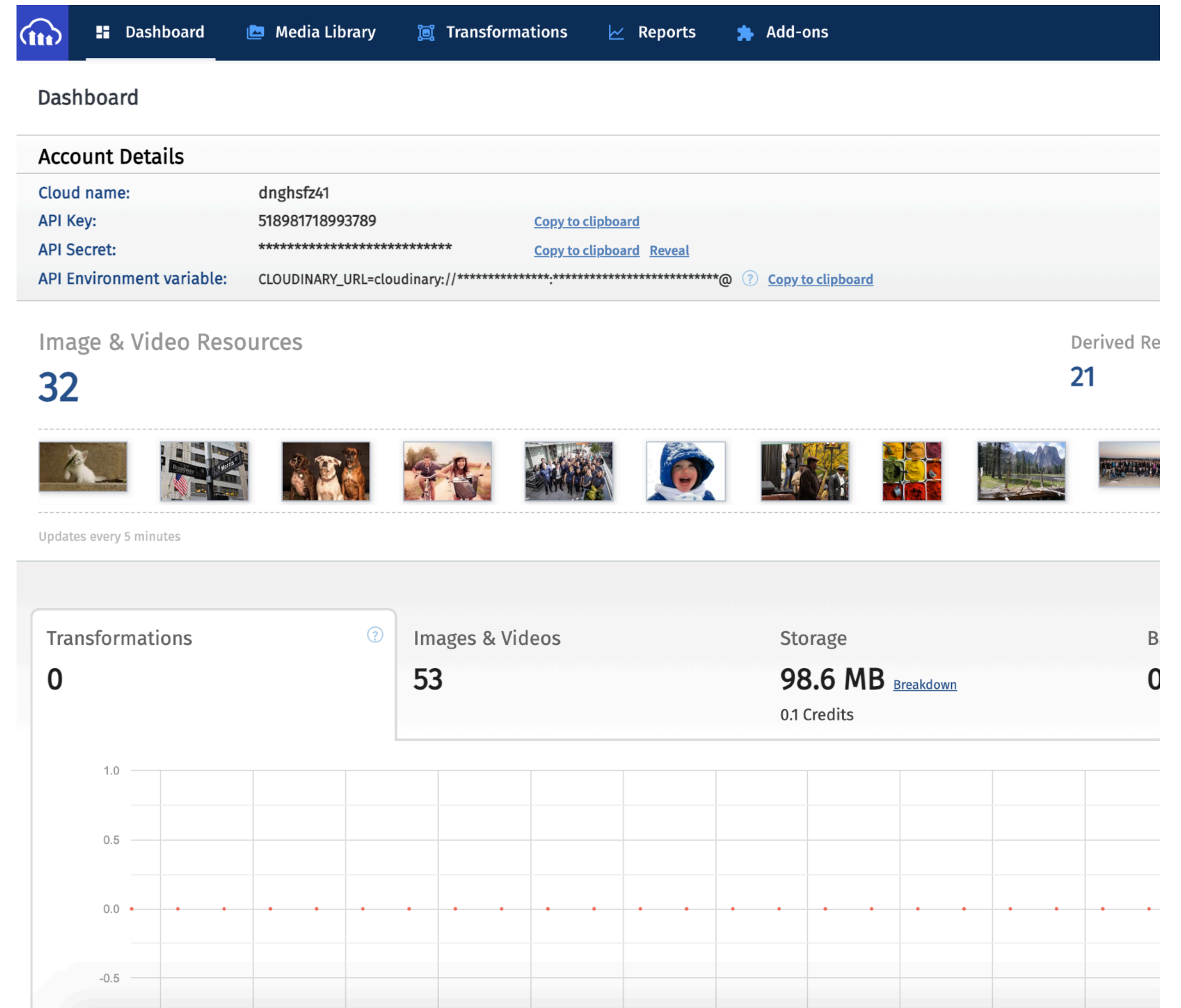
- 서버용량에 부담을 줄수있는 고용량의 이미지 파일만 따로 저장할 수 있게끔 만들어 놓은 데이터베이스.
- 그 용도에 맞게 상당히 가볍게 사용할 수 있음.



Cloudinary

Image database

- 1GB 정도의 저장공간을 무료로 사용
- 내가 가볍게 사용한건지는 몰라도 특별히 복잡하거나 무겁다고 느껴지지 않았음.
- Post,Get 되는 이미지 파일들을 규격의 맞게 설정을 해 데이터베이스를 관리할수있다는게 매력적이었음.



마치며

드림코딩 엘리 리엑트 클론코딩 - 명함만들기 Web

느낀점

- 파이어베이스를 사용하면 시간을 상당히 단축시켜 프로젝트를 만들수있다. 하지만 의존성이 너무 강해지면 종속된다는거 자체가 불안하기 때문에 그냥 한번 써본걸로 만족하고 좀더 근본적인 내 실력을 키우는게 더 먼저인거같다.
- 클라우드너리는 이미지 파일만 데이터베이스에 보관하기 위해 사용했다. 나름 많이 사용할꺼같다. 의존성도 깊지않고 굳이 클라우디너리가 아니어도 다른방법은 많을거 같다.
- 항상 프로젝트를 하면서 느끼지만 바닐라 자바스크립트를 잘 해놔야 진행에 있어서, 혹은 스스로 있어서 좀더 자신감을 갖고 할수있을거같다. 쉬우면서도 깊게 파면 어려운 자바스크립트.
- 리엑트는 충분히 강력한 만큼 많은 시간을 써야할거같다.