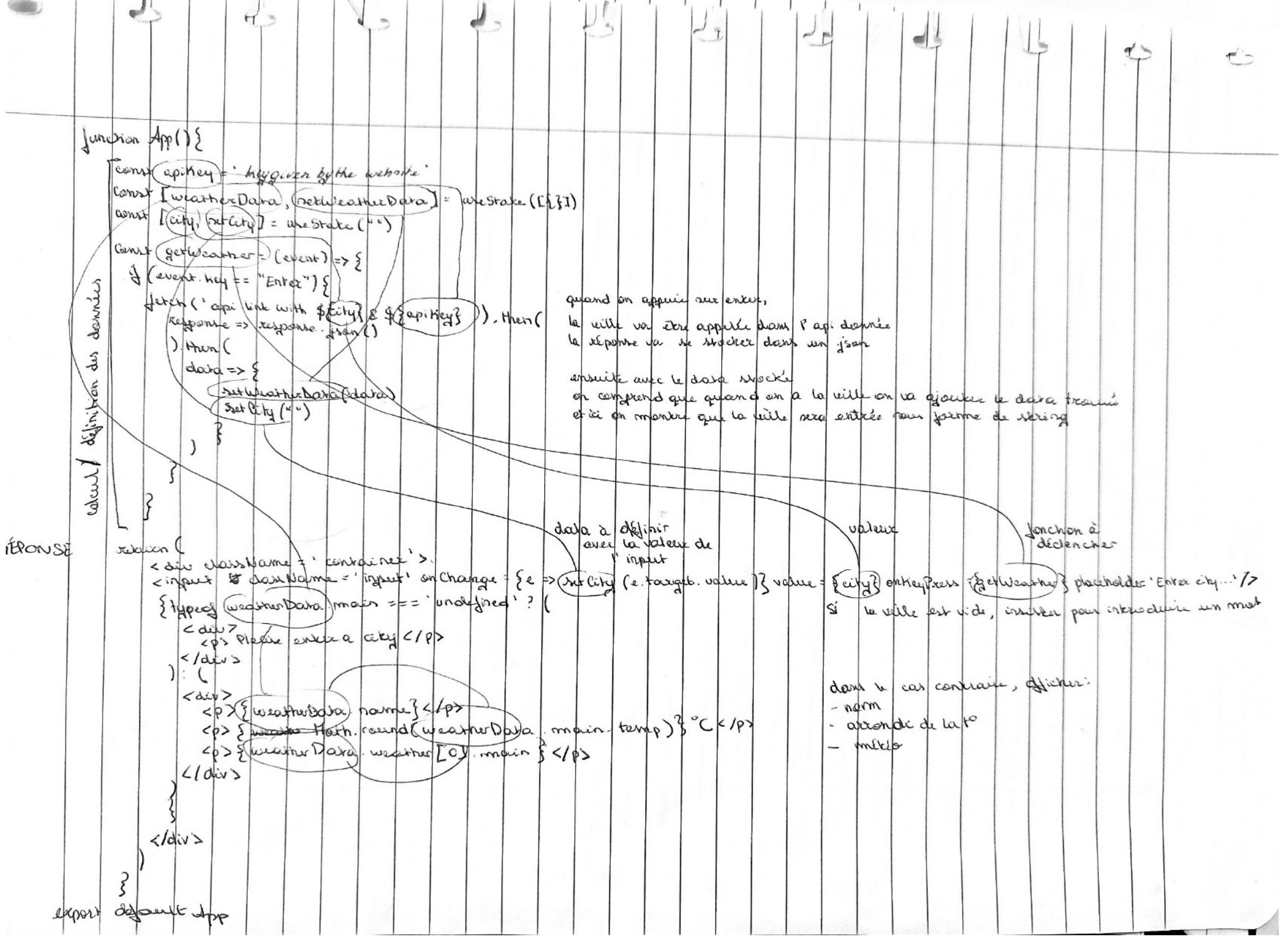
calculs : définit° des données

1. créer une fonction
2. définir la clé de l'API qui va permettre d'utiliser l'API
3. définir une constante où on va stocker les données
4. définir une constante où on va stocker la ville
5. créer un événement où lorsqu'on clique sur enter, on va chercher les données météo de la ville entrée
6. stocker les réponses de l'API dans les constantes définies plus ha
7. HTML inclure les réponses attendues (nom, température et météo)
8. définir un texte par défaut si rien n'est entré


function App(){
const apiKey = 'key given by the website'
const [weatherData, setWeatherData] = useState([{}])
const [city, setCity] = useState("")
const getWeather = (event) => {
if (event.key == "Enter") {
fetch('api link with ${city} & ${apiKey}').then(
reponse => reponse.json()
).then(
data => {
setWeatherData(data)
setCity("")
}
)
}
}
Calcul / définition des données
quand on appuie sur enter,
la ville va être appelée dans l'api donnée
la réponse va se stocker dans un json
ensuite avec le data stocké
on comprend que quand on a la ville on va ajouter le data trouvé
et ici on montre que la ville sera entrée sous forme de string
data à définir avec la valeur de l'input
valeur
fonction à déclencher
RÉPONSE
return (

<input type='text' className='input' onChange={e => setCity(e.target.value)} value={city} onKeyPress={getWeather} placeholder='Enter city...' />
Si la ville est vide, invite à pour introduire un mot
{typeof weatherData.main === 'undefined' ? (

Please enter a city

) : (

{weatherData.name}
{Math.round(weatherData.main.temp)} °C
{weatherData.weather[0].main}

)}

)
}
dans le cas contraire, afficher :
- nom
- arrondi de la t°
- météo
export default App

```
function App() {
  const [currentWeather, setCurrentWeather] = useState(null);
  const [forecast, setForecast] = useState(null);
  const handleOnSearchChange = (searchData) => {
    const [lat, lon] = searchData.value.split(" ");
    const currentWeatherFetch = fetch(`${...weather`)
    const forecastWeatherFetch = fetch(`${...forecast`)
    Promise.all([currentWeatherFetch, forecastWeatherFetch])
      .then(async (response) => {
        const weatherResponse = await response[0].json();
        const forecastResponse = await response[1].json();
        setCurrentWeather({ city: searchData.label, ...weatherResponse });
        setForecast({ city: searchData.label, ...forecastResponse });
      })}
  return (
    <div>
      <Search onSearchChange={handleOnSearchChange} />
      {currentWeather && <CurrentWeather data={currentWeather} />}
      {forecast && <Forecast data={forecast} />}
    </div>
  )}
```

{ fetch les données et les ranger

récupérer les données du currentWeatherFetch et les stocker dans un response.json

" " " " forecastWeatherFetch " " " " " "

définir setCurrentWeather en récupérant la ville et les données

" setForecast " " " " " "

constante qui se déclenche lorsqu'un changement est observé au niveau de search

en théorie pour unsplash :

```
const [unsplashImage, setUnsplashImage] = useState( );
  const unsplashFetch = fetch(...)
  Promise.all : rajouter unsplashFetch
    .then(async(response) => {
      const unsplashResponse = await response[2].json();   // image
      setUnsplashImage({city: searchData.label, ...unsplashResponse});
  return (                                                 // image
    {unsplashV && <Unsplash data={unsplashImage} />}
                  ↓
                après
```

YouTube resources:

- How to Build a Weather App with React
- React JS Tutorial – Build a Weather App With Cities Autocomplete