

REACT

Qu'est-ce que React?

- React est une bibliothèque JavaScript - l'une des plus populaires, avec plus de 100 000 étoiles sur GitHub .
- React n'est pas un framework (contrairement à Angular, qui est plus opiniâtre).
- React est un projet open-source créé par Facebook.
- React est utilisé pour créer des interfaces utilisateur (UI) sur le front-end.
- React est la couche de vue d'une application MVC (Model View Controller)

Facebook met à la cli de react 'CREATE REACT APP', un environnement pré-configuré avec tout ce dont vous avez besoin pour créer une application React. Il créera un serveur de développement en direct, utilisera Webpack pour compiler automatiquement React, JSX et ES6, prefixera automatiquement les fichiers CSS et utilisera ESLint pour tester et avertir des erreurs dans le code.

Pour configurer *create-react-app*, exécutez le code suivant dans votre terminal.

Avec npm :

```
npm init react-app my-app
```

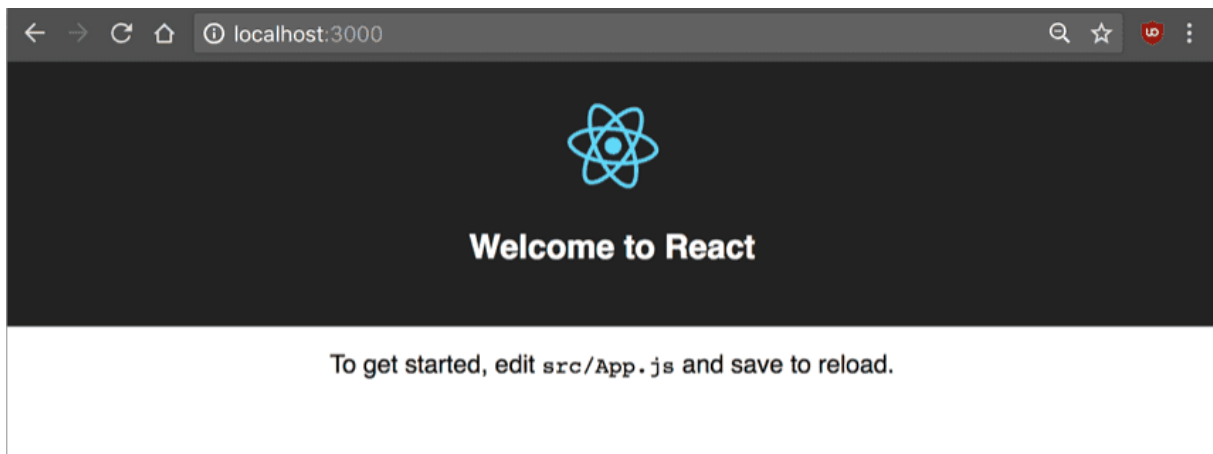
Ou avec yarn

```
yarn create react-app my-app
```

Une fois l'installation terminée, accédez au répertoire nouvellement créé et démarrez le projet.

Faites `cd my-app` et `npm start` ou `yarn start`

Une fois que vous exécutez cette commande, une nouvelle fenêtre apparaîtra *localhost:3000* avec votre nouvelle application React.



Si vous regardez dans la structure du projet, vous verrez un `/public` et `/src` répertoire, ainsi que les réguliers `node_modules`, `.gitignore`, `README.md` et `package.json`.

Votre application sera développée dans le dossier `src`.

Pour commencer :

Ajoutons les liens bootstrap ainsi que font-awesome pour bénéficier de la mise en page, mise en forme ainsi que les icônes dans `index.html`.

```
public > index.html > html > head > script
24     work correctly both with client-side routing and a non-root public URL.
25     Learn how to configure a non-root public URL by running `npm run build`.
26     -->
27     <meta name="viewport" content="width=device-width, initial-scale=1">
28     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/c
29
30     <title>Concessionnaire</title>
31     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
32     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper
33     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.
34     <script src="https://kit.fontawesome.com/8b41a163ea.js"></script>
35 </head>
```

Mettons d'abord en place notre barre de navigation :

Lançons la commande

```
npm install react-router-dom.js.
```

Créons un dossier `components` dans `src` pour stocker nos composants.

Pour gérer notre barre de navigation, mettons en place un composant dont le nom du fichier est : `Menu.js`

Notre barre de navigation gèrera trois composants : Accueil, Admin et la page 404.

Configurons notre menu en important dans `Menu.js` les différents modules de config de `react-router-dom` ainsi que nos trois composants.

```

JS Menujs X JS Accueil.js JS Page404.js JS Admin.js JS index.js JS Ajout.js JS Liste.js <> index.html
src > components > JS Menujs > Menu
1 import React from 'react';
2 import { BrowserRouter as Router, Route, Switch, Link } from 'react-router-dom';
3 import Accueil from './Accueil';
4 import Admin from './Admin';
5 import Page404 from './Page404';
6
7 const Menu = () => {
8   return (

```

Le code jsx de la barre de navigation :

```

8   return (
9     <Router>
10    <nav className="navbar navbar-expand-lg navbar navbar-dark bg-primary">
11      <Link className="navbar-brand" href="#">Navbar </Link>
12      <button className="navbar-toggler" type="button" data-toggle="collapse" da
13        <span className="navbar-toggler-icon"></span>
14    </button>
15    <div className="collapse navbar-collapse" id="navbarText">
16      <ul className="navbar-nav mr-auto">
17        <li className="nav-item active">
18          <Link className="nav-link" to="/">Accueil <span className="sr-only
19        </li>
20      </ul>
21      <span className="navbar-text">
22        <Link to="/admin123">Adminstration</Link>
23      </span>
24    </div>
25    </nav>

```

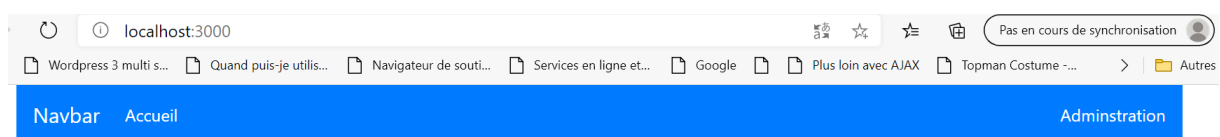
La table de routage ☹️ (En fonction de la route spécifiée , le composant correspondant sera exécutée.

```

26    <Switch>
27      <Route exact path="/">
28        <Accueil />
29      </Route>
30      <Route path="/admin123">
31        <Admin />
32      </Route>
33      <Route path="*">
34        <Page404 />
35      </Route>
36    </Switch>
37  </Router>

```

L'affichage :



Page d'accueil

Pour notre développement, nous commencerons par la partie administration :

Et développons la fonctionnalité de listage de données , déclarons un jeu de données stocké dans le state du composant admin comme suit :

```
JS App.js JS Menu.js JS Accueil.js JS Page404.js JS Admin.js X JS index.js JS Ajout.js JS Liste.js < index.html
src > components > JS Admin.js > ...
1 import React, {Component, Fragment} from 'react';
2
3 class Admin extends Component {
4   state = {
5     datas:[
6       {id:1, marque:'Peugeot', modele:'208', pays:'France', image:'208.jpg'},
7       {id:2, marque:'Renault', modele:'Clio4', pays:'France', image:'clio4.jpg'},
8       {id:3, marque:'Tesla', modele:'X', pays:'Usa', image:'x.jpg'},
9       {id:4, marque:'Bmw', modele:'M5', pays:'Allemagne', image:'m5.jpg'},
10    ]
11  }
```

Créons un composant de fonction pour parcourir notre tableau de données. Nous nommerons ce composant par Liste.

Nous transmettons les données du composant Admin vers Liste avec la propriété voitures comme suit :

```
50 render() {
51   return (
52     <Fragment>
53       <h1 className="bg-secondary text-white text-center">Administration</h1>
54       <Liste voitures = {this.state.datas} />
55     </Fragment>
56   );
57 }
58 }
59
60 export default Admin;
```

Dans Liste.js, nous récupérerons la propriété voitures venue du composant Admin via props.voitures dans le composant Liste. Nous itérerons sur props.voitures et extraire les données dans un tableau html. Ce qui donne :





```
> components > JS Liste.js > [0] Liste
1 import React from 'react';
2
3 const Liste = (props)=> {
4
5   return (
6
7     <table className="table table-striped">
8       <thead className='thead-dark'>
9         <tr>
10          <th>Id</th><th>Marque</th><th>Modele</th><th>Pays</th><th>Image</th><th>Actic
11        </tr>
12      </thead>
13      <tbody>
14        {
15          props.voitures.map((voiture, index) =>{
16            return(
17              <tr key={index}>
18                <td>{voiture.id}</td>
19                <td>{voiture.marque}</td>
20                <td>{voiture.modele}</td>
21                <td>{voiture.pays}</td>
```

```

17         <tr key={index}>
18             <td>{voiture.id}</td>
19             <td>{voiture.marque}</td>
20             <td>{voiture.modele}</td>
21             <td>{voiture.pays}</td>
22             <td>
23                 <img src={process.env.PUBLIC_URL + '/images/' + voiture.image} width={voiture.image} />
24             </td>
25             <td>
26                 <button className="btn btn-danger" onClick={() => {if(window.confirm('Voulez-vous supprimer cette voiture?')) {
27                     <i className="fa fa-trash"> Supprimer</i>
28                 }}>
29             </td>
30         </tr>
31     </tr>
32     )
33 })
34
35 </tbody>
36

```

A l'exécution, nous avons :

Id	Marque	Modele	Pays	Image	Action
1	Peugeot	208	France		<button>Supprimer</button>
2	Renault	Clio4	France		<button>Supprimer</button>
3	Tesla	X	Usa		<button>Supprimer</button>
4	Bmw	M5	Allemagne		<button>Supprimer</button>

Intéressons nous maintenant à la fonctionnalité de suppression :

Pour ce faire ajoutons une méthode de suppression nommée `removeVoiture()` cette méthode prendra en paramètre l'indice de l'élément à supprimer.

Utilisons la méthode `Array.filter()` qui crée et retourne un nouveau tableau contenant tous les éléments du tableau d'origine qui remplissent une condition déterminée par la fonction callback.

Notre condition ici pour avoir le nouveau tableau sera tous les éléments du state sauf celui dont l'index est égal à l'index passé en paramètre de la méthode. Après mettons à jour notre state.

Ce qui donne :

```

28     removeVoiture = (index) => {
29         const {datas} = this.state
30         this.setState({
31             datas: datas.filter((d, i) => {
32                 return i !== index
33             })
34         });
35     };
36
37 }
38

```

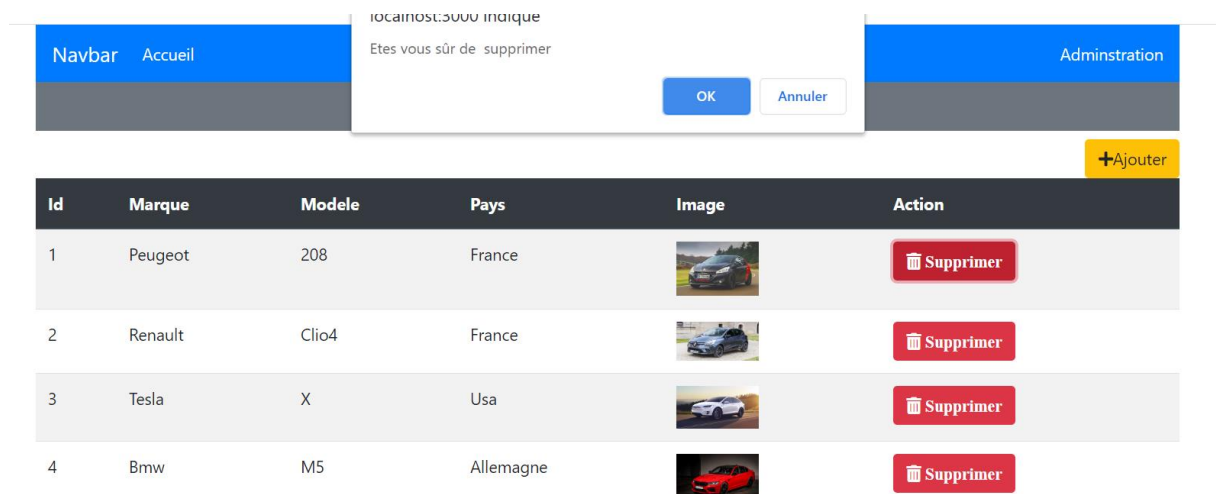
Pour exécuter la suppression nous aurons besoin de renseigner l'index du paramètre. Cette info viendra du composant Liste, car c'est dans ce composant que l'utilisateur choisira l'élément à supprimer.

Alors passons alors cette méthode sous forme propriété nommée : `deleteVoiture` dans la Liste.

```
48   render() {
49     return [
50       <Fragment>
51         <h1 className="bg-secondary text-white text-center">Administration</h1> |
52         <Liste deleteVoiture = {this.removeVoiture} voitures = {this.state.datas} />
53       </Fragment>
54     ];
55   }
56 }
```

Récupérons cette propriété `deleteVoiture` dans le composant Liste pour renseigner la valeur de l'index. Ce qui donne :

```
26 <td>
27   <button className="btn btn-danger"
28     onClick={() => {if(window.confirm('Etes vous sûr de supprimer'))
29       {props.deleteVoiture(index)
30       };}
31   >>
32   <i className="fa fa-trash"> Supprimer</i>
33 </button>
34 </td>
```



Avec la méthode `confirm()` de js, nous demandons à l'utilisateur de confirmer la suppression avant que la méthode `deleteVoiture(index)` -> `removeVoiture(index)` ne s'exécute.

La fonctionnalité d'ajout :

Créons un composant avec état appelé : Ajout

Nous aurons besoin d'un formulaire d'ajout pour ajouter une nouvelle voiture. Dans la plupart des cas, pour implémenter des formulaires, nous recommandons d'utiliser des composants contrôlés. Dans un composant contrôlé, les données du formulaire sont gérées par le composant React. L'alternative est le composant non-contrôlé, où les données sont gérées par le DOM.

Nous utiliserons la méthode recommandée celle des formulaires contrôlés par les composants via le state :

Créons le formulaire.

```
59 <form>
60   <div className="form-group ">
61     <label htmlFor="marque">Marque</label>
62     <input type="text" className="form-control" id="marque" name="marque" value={marque} p
63   </div>
64
65   <div className="form-group ">
66     <label htmlFor="modele">Modèle</label>
67     <input type="text" className="form-control" id="modele" name="modele" value={modele} p
68   </div>
69
70   <div className="form-group ">
71     <label htmlFor="pays">Pays</label>
72     <input type="text" className="form-control" id="pays" name="pays" value={pays} placeho
73   </div>
74
75   <div className="form-group ">
76     <label htmlFor="image">Image</label>
77     <input type="text" className="form-control" id="image" name="image" value={image} plac
78   </div>
79
```

En destructurant notre state , champ champ du formulaire sera lié à chaque propriété du state :

```
15   render(){
16     const {marque, modele, pays, image} = this.state;
17     return(
```

L'évènement onChange permet de surveiller chaque champ et de déclencher la méthode modifier() pour récupérer la valeur saisie dans le champ et ainsi mettre à jour le state initial.

```
src > components > JS AjoutJS > JS Ajout
1  import React, { Component, Fragment } from 'react';
2
3  class Ajout extends Component {
4
5     initVoiture = {id:1, marque:"", modele:"", pays:"", image:""};
6     state = this.initVoiture;
7     modifier = (e) =>{
8       const {name, value} = e.target;
9       this.setState({[name]:value});
10    }
```

Lors du clic sur le bouton soumettre la méthode envoyer est invoquée qui a but de renseigner la nouvelle voiture à insérer en paramètre de la méthode ajouter(). Cette méthode est passée en paramètre dans le composant Ajout inclus dans Admin.

```
11   envoyer = (event) => {
12     event.preventDefault();
13     console.log(this.state);
14     this.props.ajouter(this.state);
15   }
```

```

49     return (
50       <Fragment>
51         <h1 className="bg-secondary text-white text-center">Administration</h1>
52         <Ajout ajouter = {this.addVoiture} />
53         <Liste deleteVoiture = {this.removeVoiture} voitures = {this.state.datas} />
54       </Fragment>
55     );
56   }

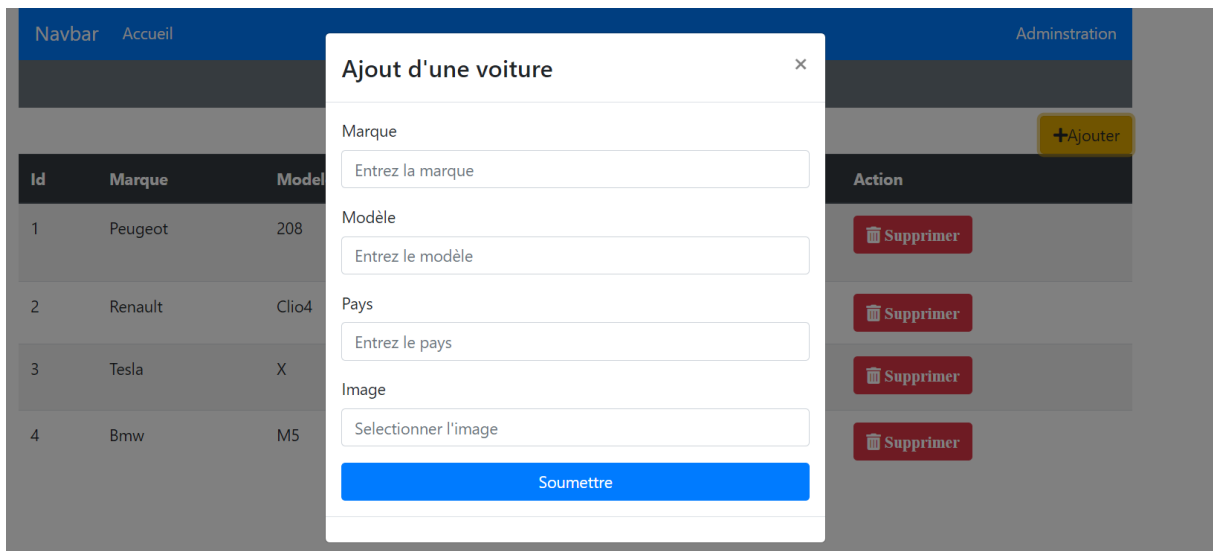
```

Et dans addVoiture() du composant Admin. On appellera notre tableau de données auquel on ajoutera la nouvelle voiture envoyée par envoyer() du composant Ajout et enfin on met ajout le state afin que l'ajout soit pris en compte . Ce qui donne :

```

38
39   addVoiture = (newVoiture) => {
40     if(this.state.datas.length !== 0){
41
42       newVoiture.id = (this.state.datas[this.state.datas.length - 1].id + 1)
43     }
44     this.setState({datas: [...this.state.datas, newVoiture]})
45   }

```



Persistons les données dans les applications de notre navigateur via le localStorage.

La syntaxe pour 'écriture de l'article localStorage est la suivante :

```
localStorage.setItem('cle', 'obj');
```

La syntaxe pour la lecture de l'article localStorage est la suivante :

```
var cat = localStorage.getItem('cle');
```

La syntaxe pour la suppression de l'élément localStorage est la suivante :

```
localStorage.removeItem('cle');
```

La syntaxe pour supprimer tous les éléments de localStorage est la suivante :


```
// Effacer tous les élément
```

Lorsqu'on veut stocker la donnée dans le localStorage et itérer dessus, on fait appel à la méthode `componentWillMount()` qui s'exécute automatiquement à l'appel du composant Admin

```
14   componentDidMount =()=>{
15     let tabVoiture = JSON.parse(localStorage.getItem('datas'));
16     if(!tabVoiture || tabVoiture.length === 0){
17       let voitures1 = localStorage.setItem('datas',JSON.stringify(this.state.datas));
18       this.setState({datas:voitures1}, function(){
19
20         console.log(this.state.datas);
21       });
22     }else{
23       let voitures2 = JSON.parse(localStorage.getItem('datas'));
24       this.setState({datas:voitures2});
25     }
26
27   }
28 }
```

A l'exécution, on a :

The screenshot shows a web application interface on the left and its developer tools on the right. The interface has a blue header with 'Administration' and a table with 4 rows of car data. The developer tools show the 'Storage' tab with a list of items in local storage, including an array of car objects.

Id	Marque	Modele	Pays	Image	Action
1	Peugeot	208	France		<button>Supprimer</button>
2	Renault	Clio4	France		<button>Supprimer</button>
3	Tesla	X	Usa		<button>Supprimer</button>
4	Bmw	M5	Allemagne		<button>Supprimer</button>

Storage

Key	Value
datas	[{"id":1,"marque": "...

Local Storage

- 0: {id: 1, marque: "Peugeot", modele: "208", pays: "France", image: "https://..."}
1: {id: 2, marque: "Renault", modele: "Clio4", pays: "France", image: "https://..."}
2: {id: 3, marque: "Tesla", modele: "X", pays: "Usa", image: "https://..."}
3: {id: 4, marque: "Bmw", modele: "M5", pays: "Allemagne", image: "https://..."}

Pour l'ajout et la suppression, nous avons successivement les méthodes :

```
38
39   addVoiture = (newVoiture) => {
40     if(this.state.datas.length !== 0){
41       newVoiture.id = (this.state.datas[this.state.datas.length - 1].id + 1)
42     }
43     this.setState({datas: [...this.state.datas, newVoiture]},function(){
44       localStorage.setItem('datas',JSON.stringify(this.state.datas));
45     })
46   }
47 }
48 }
```

```

28  removeVoiture = (index)=>{
29      const {datas} = this.state
30      this.setState({
31          datas: datas.filter((d, i) =>{
32              return i!==index
33          })
34      }, function(){
35          localStorage.setItem('datas',JSON.stringify(this.state.datas));
36      })
37  }

```

Etudions comment consommer une web api avec React.

Prenons l'exemple de l'api : openweathermap

<https://openweathermap.org/>

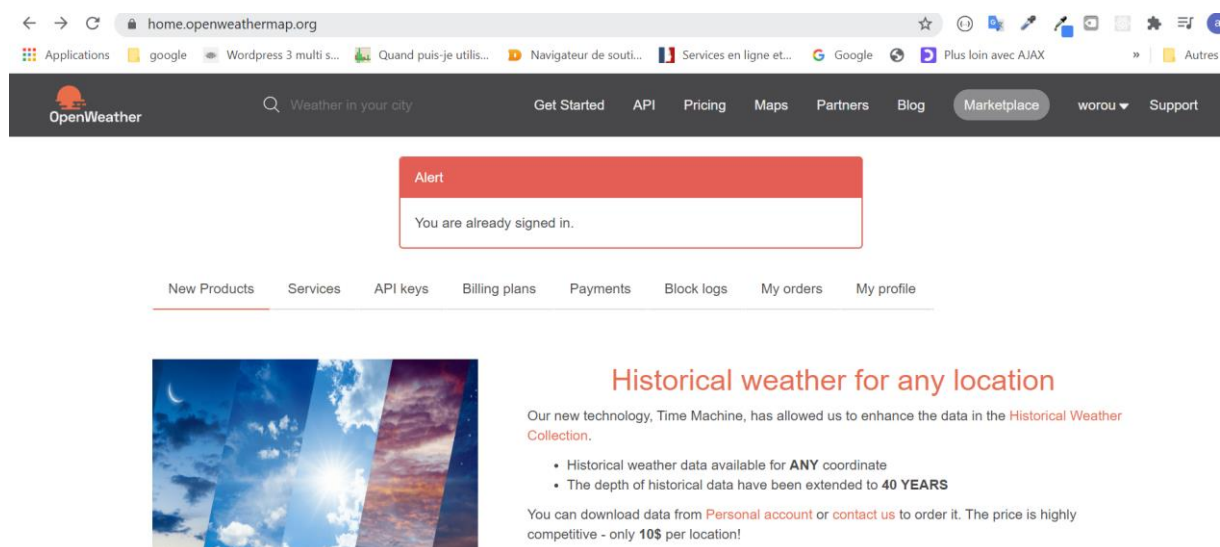
<https://openweathermap.org/weather-conditions>

OpenWeatherMap est l'un des principaux fournisseurs d'informations météorologiques numériques. ils étaient une petite société informatique, créée en 2014 par un groupe d'ingénieurs et d'experts en Big Data, traitement de données et traitement d'images satellites. Leur siège social est au Royaume-Uni, ils ont un bureau aux États-Unis et l'équipe de développement en Lettonie (UE).

Ils comptent plus de 2 000 000 d'utilisateurs de leurs produits et reçoivent environ 2 000 nouveaux utilisateurs chaque jour. Ils sont partenaires de Google qui utilisent l'API météo d'OpenWeatherMap dans leurs AdWords pour la gestion des campagnes publicitaires.

L'utilisation de l'API OpenWeatherMap est gratuite ; cependant, il existe également des options payantes pour les développeurs qui ont besoin de plus de données météorologiques et d'assistance.

Pour utiliser l'api, vous aurez besoin de générer votre clé.



The screenshot shows the OpenWeatherMap website. At the top, there's a navigation bar with links like 'Get Started', 'API', 'Pricing', 'Maps', 'Partners', 'Blog', 'Marketplace', 'worou', and 'Support'. Below the navigation bar, there's a search bar with the text 'Weather in your city'. A red alert box says 'Alert: You are already signed in.' Below the alert, there's a horizontal menu with links: 'New Products', 'Services', 'API keys', 'Billing plans', 'Payments', 'Block logs', 'My orders', and 'My profile'. The main content area features a large image of a sunset/sunrise over a body of water. To the right of the image, there's a section titled 'Historical weather for any location'. The text below the title says: 'Our new technology, Time Machine, has allowed us to enhance the data in the Historical Weather Collection.' Below this, there are two bullet points: '• Historical weather data available for ANY coordinate' and '• The depth of historical data have been extended to 40 YEARS'. At the bottom, it says: 'You can download data from Personal account or contact us to order it. The price is highly competitive - only 10\$ per location!'.

[New Products](#) [Services](#) [API keys](#) [Billing plans](#) [Payments](#) [Block logs](#) [My orders](#) [My profile](#)

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	
d803cb20c3fe8bceab994217320ce0bc	Default	✎ ✕
74da1b43c92202818ef1c14835e86054	alt	✎ ✕

Create key

API key name

Generate

If you have some problem with activation your billing subscription please write to [support center](#)

Current weather

[See details](#)

Name	Calls per minute (no more than)	Hourly forecast (days)	3 hour forecast (days)	Daily forecast (days)	Price per month (excl. VAT)	Status
Free plan	60	unavailable	5	unavailable	Free	Default
Startup plan	600	unavailable	5	16	40 USD	Subscribe
Developer plan	3000	4.5	5	16	180 USD	Subscribe
Professional plan	30000	4.5	5	16	470 USD	Subscribe
Enterprise plan	200000	4.5	5	16	2000 USD	Subscribe

Historical data

[See details](#)