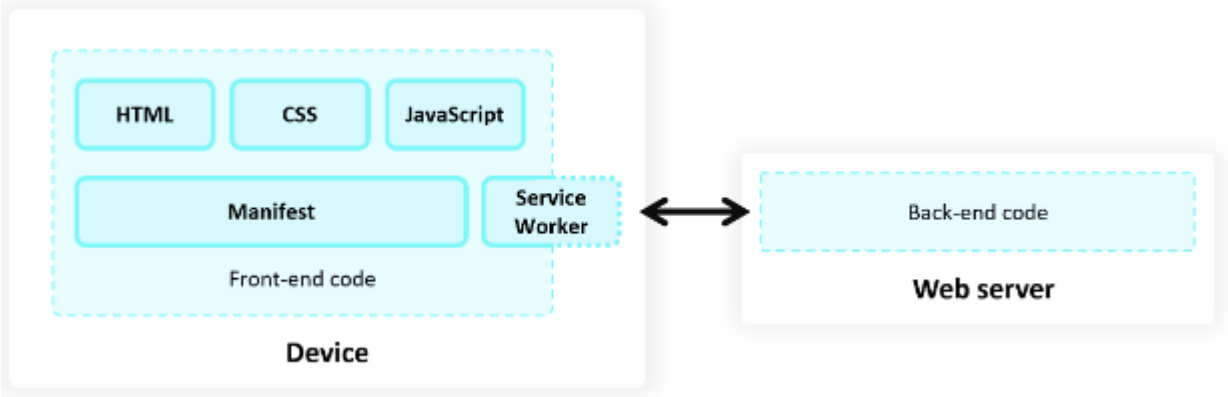
ARCHITECTURE :

Le code frontal utilise uniquement HTML, CSS, JavaScript et un manifeste JSON.

Le code back-end peut utiliser les langages côté serveur de votre choix tels que ASP.NET, Java, Node.js ou PHP. Ici, le pwa ne contient aucun code côté serveur, car l'application s'exécute exclusivement sur l'appareil sur lequel elle est installée et n'a besoin d'aucune donnée côté serveur.



DEMARRAGE :

- Créer un nouveau dossier sur votre ordinateur dans lequel le serveur Web s'exécutera.
- Démarrez le serveur à l'aide de la http-server : `> npx http-server`
- Exécuter un simple serveur Web local : `http://localhost:8080`

PAGE HTML :

```
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="convertir.css">
    <link rel="manifest" href="/manifest.json">
    <title>Convertisseur de température</title>
  </head>
body>
  <h1>Convertisseur de température</h1>
  <form id="converter">
      <label for="input-temp">temperature</label>
      <input type="text" id="input-temp" name="input-temp" value="20" />
      <label for="input-unit"></label>
      <select id="input-unit" name="input-unit">
        <option value="c" selected>Celsius</option>
        <option value="f">Fahrenheit</option>
        <option value="k">Kelvin</option>
      </select>
      <label for="output-unit">vers</label>
      <select id="output-unit" name="output-unit">
        <option value="c">Celsius</option>
        <option value="f" selected>Fahrenheit</option>
        <option value="k">Kelvin</option>
      </select>
      <output name="output-temp" id="output-temp" for="input-temp input-unit output-unit">68 F</output>
    </form>
```

```
    <script src="convertir.js"></script>
    <script>
      if('serviceWorker' in navigator) {
        navigator.serviceWorker.register('/sw.js', { scope: '/' });
      }
    </script>
</body>
```

MANIFESTE

manifest.json > ...

```
{
    "lang": "en-us",
    "name": "Convertisseur de température",
    "short_name": "CTA",
    "description": "conversion d'une température en Celsius, Kelvin, et Fahrenheit",
    "start_url": "/",
    "background_color": "#2f3d58",
    "theme_color": "#2f3d58",
    "orientation": "any",
    "display": "standalone",
    "icons": [
        {
            "src": "/icon512.png",
            "sizes": "512x512"
        }
    ]
}
```

STYLE :

```
#converter {
  width: 15rem;
  padding: 2rem;
  border-radius: .5rem;
  box-shadow: 0 0 2rem 0 #0001;
  display: flex;
  flex-direction: column;
  align-items: center;
}

#converter input, #converter select {
  font-family: inherit;
  font-size: inherit;
  margin-block-end: 1rem;
  text-align: center;
  width: 10rem;
}

#converter #output-temp {
  font-size: 2rem;
  font-weight: bold;
}
```

JAVASCRIPT :

```javascript
const inputField = document.getElementById('input-temp');
const fromUnitField = document.getElementById('input-unit');
const toUnitField = document.getElementById('output-unit');
const outputField = document.getElementById('output-temp');
const form = document.getElementById('converter');

function convertTemp(value, fromUnit, toUnit) {
  if (fromUnit === 'c') {
    if (toUnit === 'f') {
      return value * 9 / 5 + 32;
    } else if (toUnit === 'k') {
      return value + 273.15;
    }
    return value;
  }
  if (fromUnit === 'f') {
    if (toUnit === 'c') {
      return (value - 32) * 5 / 9;
    } else if (toUnit === 'k') {
      return (value + 459.67) * 5 / 9;
    }
    return value;
  }

  if (fromUnit === 'k') {
    if (toUnit === 'c') {
      return value - 273.15;
    } else if (toUnit === 'f') {
      return value * 9 / 5 - 459.67;
    }
    return value;
  }
  throw new Error('Invalid unit');
}

form.addEventListener('input', () => {
  const inputTemp = parseFloat(inputField.value);
  const fromUnit = fromUnitField.value;
  const toUnit = toUnitField.value;

  const outputTemp = convertTemp(inputTemp, fromUnit, toUnit);
  outputField.value = (Math.round(outputTemp * 100) / 100) + ' ' + toUnit.toUpperCase();
});
```

CODE DU SERVICE WORKER

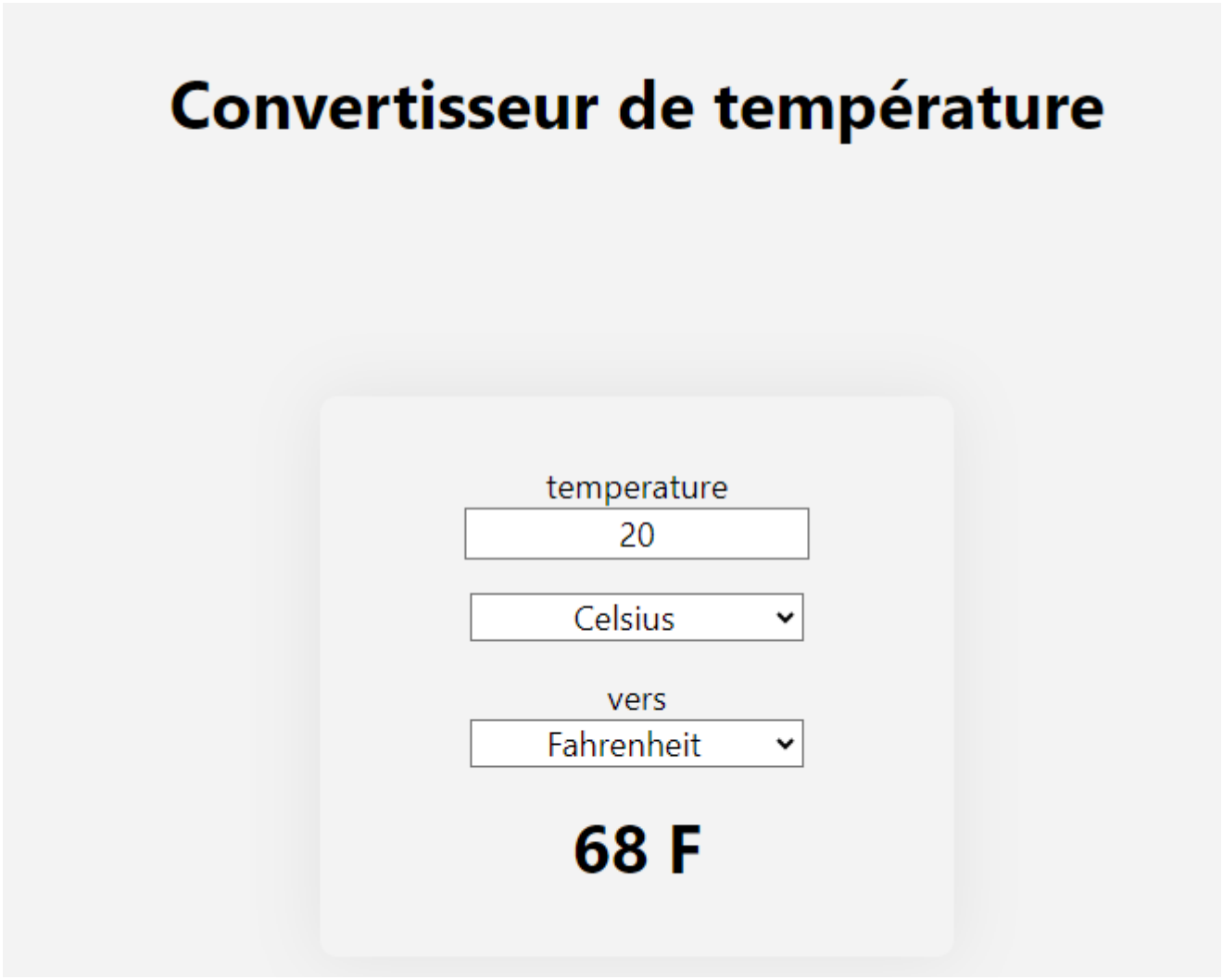
```
const CACHE_NAME = `CT-v1`;

self.addEventListener('install', event => {
  event.waitUntil((async () => {
    const cache = await caches.open(CACHE_NAME);
    cache.addAll([
      '/',
      '/convertir.js',
      '/convertir.css'
    ]);
  })());
});

self.addEventListener('fetch', event => {
  event.respondWith((async () => {
    const cache = await caches.open(CACHE_NAME);

    const cachedResponse = await cache.match(event.request);
    if (cachedResponse) {
      return cachedResponse;
    } else {
        try {
          const fetchResponse = await fetch(event.request);

          cache.put(event.request, fetchResponse.clone());
          return fetchResponse;
        } catch (e) {
          // The network failed.
        }
    }
  })());
});
```

Ce code écoute l'événement install et l'utilise pour mettre en cache toutes les ressources dont l'application a besoin pour fonctionner : la page HTML de démarrage, le fichier JavaScript du convertisseur et le fichier CSS du convertisseur.

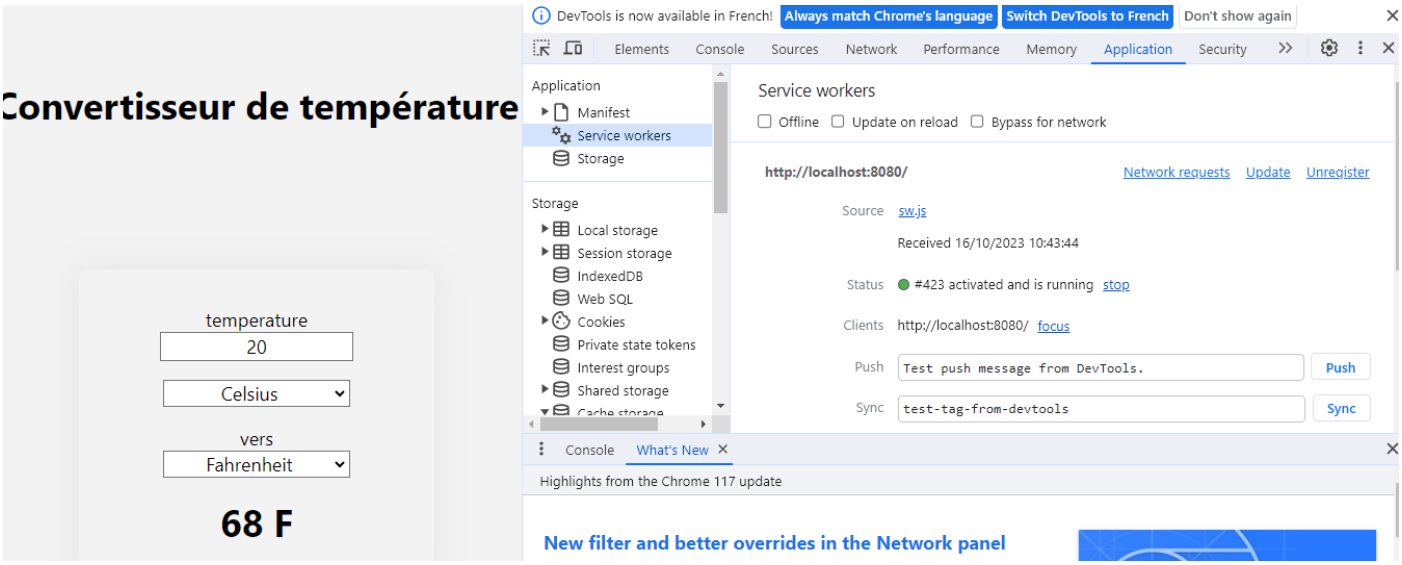
Ce code intercepte également les événements fetch qui se produisent chaque fois que l’application envoie une requête au serveur et applique une stratégie de priorité au cache. Le service worker renvoie les ressources mises en cache afin que l’application puisse fonctionner hors ligne et, si cela échoue, tente de télécharger à partir du serveur.

LANCEMENT

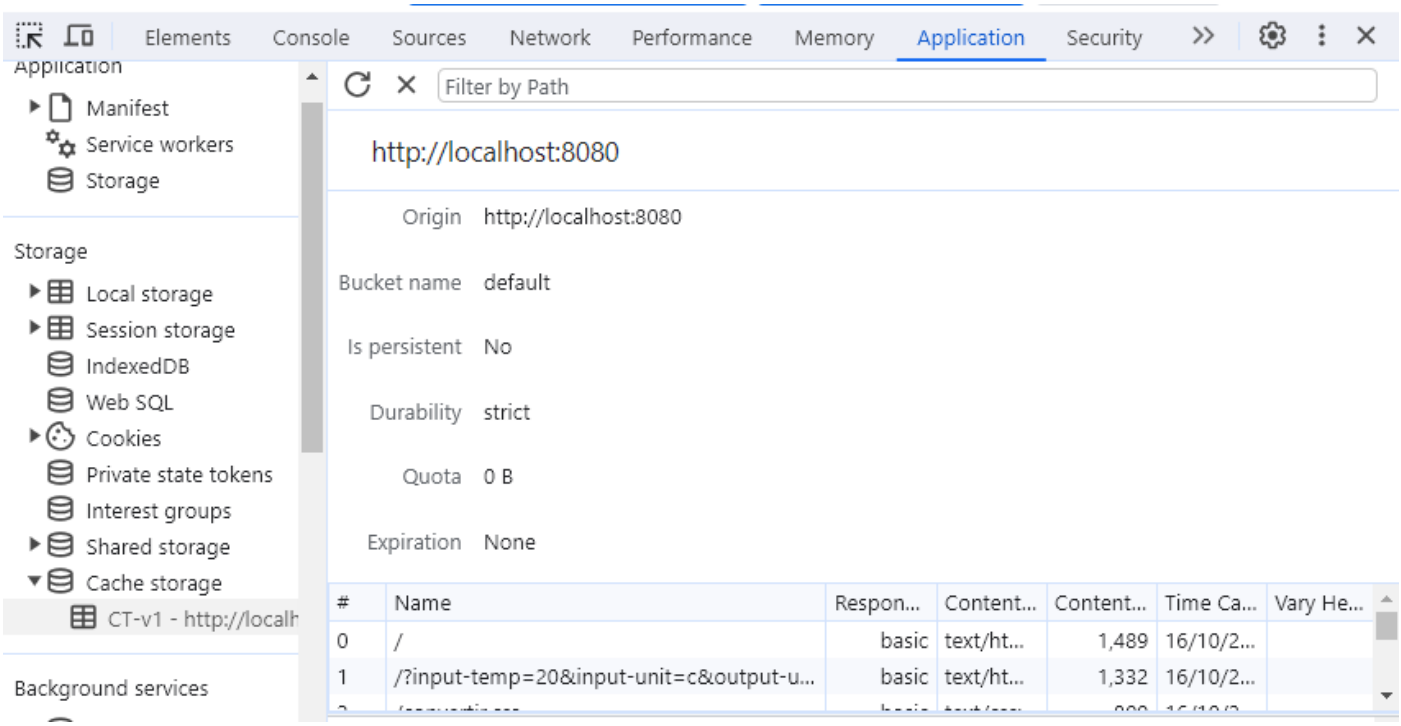
1. Dans un navigateur, accédez à http://localhost:8080.



2. Pour ouvrir DevTools, choisir **Inspecter** Ou appuyez sur **Ctrl+Maj+I** (Windows, Linux) ou **Commande+Option+I** (macOS). DevTools s'ouvre.
3. Ouvrez l' outil **Application** , puis **Service Workers** . Si le service worker n'est pas affiché, actualisez la page.



Affichez le cache du service worker en développant **Cache Storage** et en sélectionnant **CT-v1** . Toutes les ressources mises en cache par le service worker doivent être affichées. Les ressources mises en cache par le service worker incluent l'icône de l'application, le manifeste de l'application et la page initiale.



INSTALLATION

Maintenant que l'application dispose d'un manifeste d'application Web et d'un service worker, les navigateurs pris en charge peuvent l'installer en tant que PWA.

Dans un navigateur, une fois que l'application est actualisée, le bouton **Application disponible** apparaît dans la barre d'adresse. En cliquant sur le bouton **Application disponible** , vous êtes invité à installer l'application localement.

Cliquez sur **Installer** pour installer l'application localement. Une fois l'installation terminée, votre application s'affiche dans sa propre fenêtre et sa propre icône d'application dans la barre des tâches.