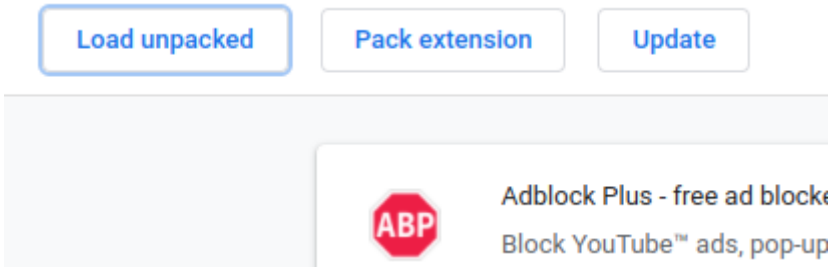
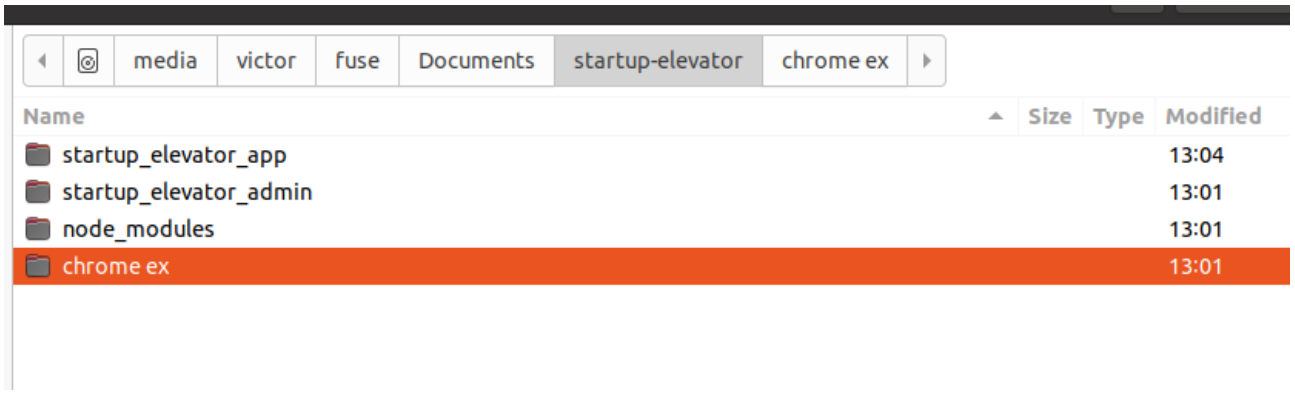
Extension chrome avec API REST

Le projet consiste en une extension chrome qui communique avec une API Rest sur un site fait avec meteor pour y envoyer et pour récupérer des données.

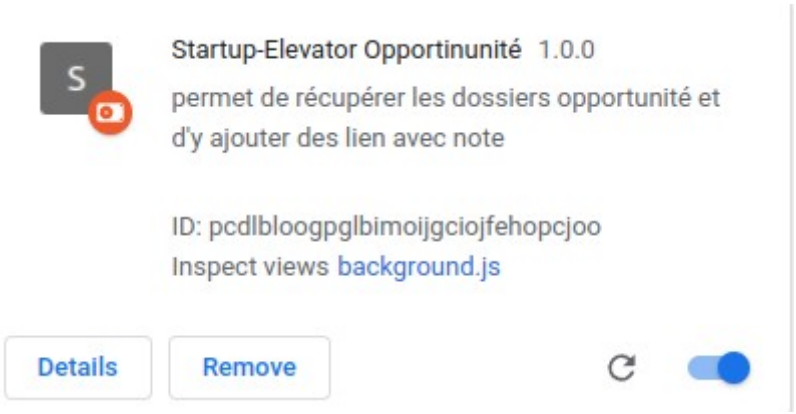
I l’extension chrome

pour l’installer, il vous faudra récupérer le dossier chrome ex, sélectionner l’option Load unpacked dans chrome://extensions puis sélectionner le dossier.





L’extension s’appelle Startup-Elevator Opportunité ces informations sont présentes dans le manifest.json qui donne tout les informations titre logo autorisation, sur quel site l’extension peut fonctionner etc...



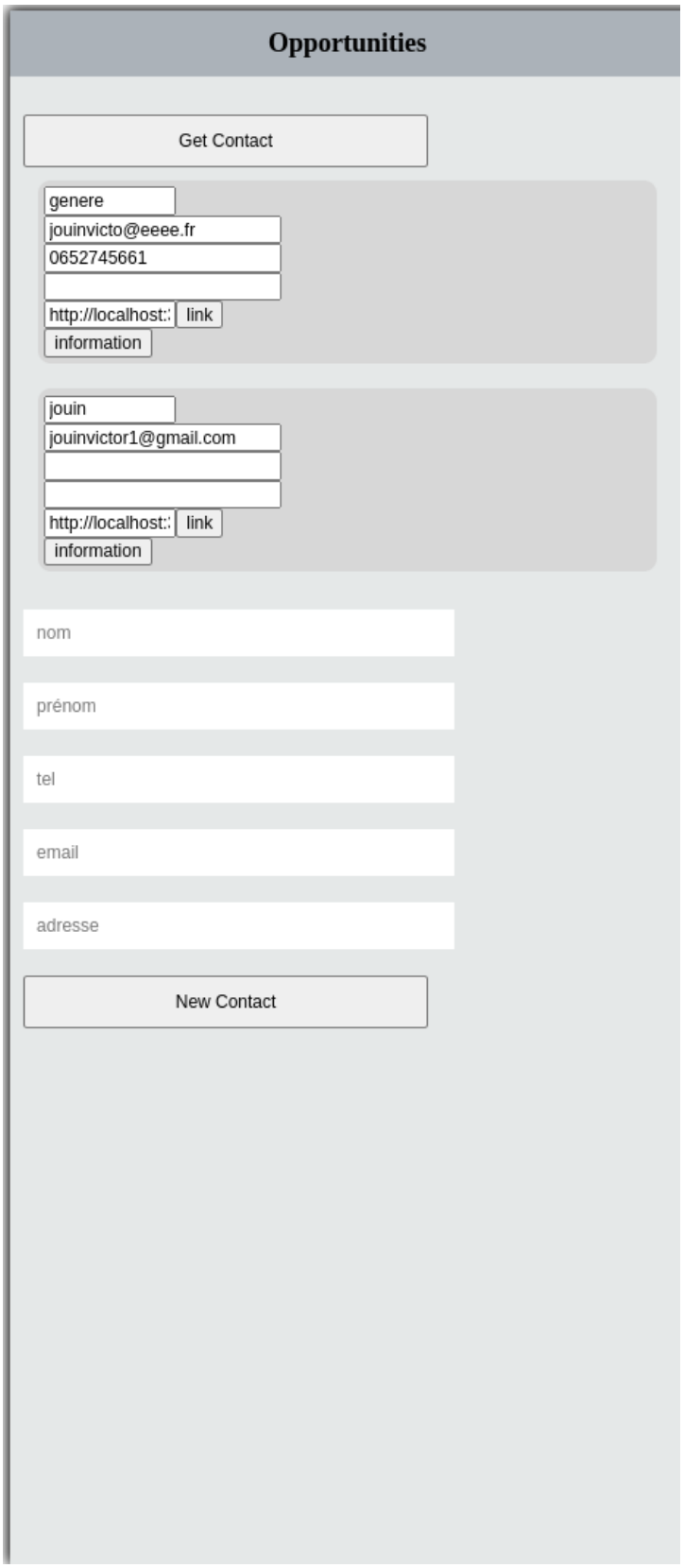
le manifest.json

```
    "manifest_version": 2,
    "name": "Startup-Elevator Opportinunité",
    "description": "permet de récupérer les dossiers opportunité et d'y ajouter des lien avec
note",
    "version": "1.0.0",
    "content_security_policy":"script-src 'self' https://apis.google.com; object-src 'self'",
    "browser_action": {
        "default_popup": "popup1.html",
        "chrome_style": false
    },
    "content_scripts": [
        {
        "matches": ["<all_urls>"],
        "js": ["jquery-3.6.0.min.js","content.js" ],
        "presistent": "true",
        "css":        ["style.css"]
    }
    ],
    "background":
    {
        "page": "background.js"
    },
    "permissions": [
        "https://startup-elevator.com/",
        "http://localhost:4000/*",
        "http://localhost/*",
        "declarativeContent",
        "activeTab",
        "tabs",
        "storage"
        ]
}
```

L’extension ce présente ainsi:

elle a un bouton pour créer des utilisateurs et un autre qui va les récupérer sur l’API
du site

celle si a aussi une option de link,
le lien de la page est récupéré et peut être envoyé sur n’importe quelle utilisateur récupéré par L’API.



L’extension utilise Jquery pour pouvoir avoir une mise à jour des ressources sans avoir besoin de recharger la page

```
var script = document.createElement('script');
script.src = 'https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js'
script.type = 'text/javascript';
document.getElementsByTagName('head')[0].appendChild(script);
//div.attachShadow({ mode: 'open' }).innerHTML +=

htmlme();
showdata();
//showdatalink();
```

les ressources html sont générées avec du js j’ai essayé de faire ça avec un shadowDOM pour que le CSS des pages n’agissent pas avec l’extension mais cela bloquai certaine fonctionnalité, je travail toujours dessus

```
// extension display

function htmlme() {
  var div = document.createElement("div");
  document.body.appendChild(div);
  div.name="extension";
  div.innerHTML +=
    '<div id="base" name="divy" style="flex-direction: column; overflow-y: scroll; top: 9%;backface-visibility: inherit;left: 70%;posi
    '<div align="center" style="background: #ABB2B9">' +
    '<h1 style="font-size: 20px;">Opportunities</h1>' +
    '</div>' +
    '<br>' +
    '<form id="formula5" method="PUT" target="_blank" action="http://localhost:3000/projectsA7465D84desrefee7e9e86">' +
    '<button type="submit" id="submit1" style="width: 60%; padding: 10px; margin: 10px;">Get Contact</button>' +
    '</form>' +
    '<div align="center" id="contenu" style="list-style: none;align="center";width:80%">' + '</div>' +
    '<form id="formula" method="GET" target="_blank" action="http://localhost:3000/projectsA7465D84desrefee7e9e86?name="name&firstname=
    '<div id="div_form" style="">' +
    '<input name="name" type="text" style="width: 60%; border: 0px; padding: 10px; margin: 10px;" placeholder = "nom">' +
    '<input name="firstname" type="text" style="width: 60%; border: 0px; padding: 10px; margin: 10px;" placeholder = "prénom">' +
    '<input name="tel" type="text" style="width: 60%; border: 0px; padding: 10px; margin: 10px;" placeholder = "tel">' +
    '<input name="email" type="text" style="width: 60%; border: 0px; padding: 10px; margin: 10px;" placeholder = "email">' +
    '<input name="adresse" type="text" style="width: 60%; border: 0px; padding: 10px; margin: 10px;" placeholder = "adresse">' +
    '<button type="submit" id="submit" style="width: 60%; padding: 10px; margin: 10px;display: block">New Contact</button> ' +
    '</div>' +
    '</form>' +
    '</div>';
```

pour récupérer url à envoyer sur les utilisateurs voulu

```
// url

var currentPage = location.href;
```

vérification du changement d’url.

```
// vérifier si l'url change


setInterval(function () {
  if (currentPage != location.href) {
    currentPage = location.href;
    $('[name="url"]').val(window.location.href);
  }
}, 500);
}
```

la fonction qui permet de récupérer les données utilisateur via l’API
il va chercher sa promesse sur l’adresse de l’Api puis enregistrer le résultat (la réponse intitulé response.json) sur usersData.

```
// récupération donnée API project

const affichage = document.getElementsByName("test");
const promise01 = fetch("http://localhost:3000/projectsA7465D84desrefee7e9e86");
result = false;
promise01
  .then((response) => {
    console.log(response);

    const usersData = response.json();
    console.log(usersData);
    usersData.then((response) => {
      loadparam(response);
    });
  })
```

Loadparam qui est appelé permet de récupérer le nombre d’élément (Document Json)

```
// affichage des données sur l'extension

async function loadparam(response) {
  const name = response[1].userLastName;
  var count = response.length;
  storedata(response, count);
```

Puis la réponse est stocké dans la database chrome.storage via la fonction storedata

```
// stockage des donnée sur un BD chrome
function storedata(response, count) {
  for (var i = 0; i != count; i++) {
    chrome.storage.sync.set({ opportunity: response }, function () {
    });
  }
}
```

ajout de la suppression
de la data dans chrome storage
que ne pas avoir de donnée
non presente sur le siteweb

```
// stockage des donnée sur un BD chrome
function storedata(response, count) {

  // clean data before adding new data
  chrome.storage.local.clear(function() {
    var error = chrome.runtime.lastError;
    if (error) {
        console.error(error);
    }
});
```

j’utilise le storage local maintenant car celui-ci offre plus de mémoire

ici on défini la collection user (les utilisateur récupérés)
en temps que **opportunity**
avec:

chrome.storage.set(nom_collection: réponse_API), function () {});

Il y a le même procéder pour récupérer les liens (url) étant d’une autre collection (link) celle-ci utilise une autre promesse

```
// récupération donnée API links

const affichage1 = document.getElementsByName("test");
const promise02 = fetch("http://localhost:3000/link4d8e64AA856HGE496568efd89d86");
result = false;
promise02
  .then((response) => {
    console.log(response);

    const usersLink = response.json();
    console.log(usersLink);
    usersLink.then((response) => {
      loadparamlink(response);
    });
  })
```

```
async function loadparamlink(response) {
  const name = response[1].adresse;
  var count = response.length;
  storedatalink(response, count);
}
```

```
function storedatalink(response, count) {
  for (var i = 0; i != count; i++) {
    chrome.storage.sync.set({ links: response}, function () {
      console.log(result.links[i].adresse);
    });
  }
}
```

On arrive à l’affichage des données sur l’extension chrome

chrome.storage.sync.get(['opportunity'], function (result)) permet de récupérer la collection opportunity en nomant sa data-base résult

```
// affichage des données via la DB chrome


function showdata() {
  var count = 0;
  chrome.storage.sync.get(['opportunity'], function (result) {
    count = result.opportunity.length;
    data_display(count);
  });
}

function showdatalink() {
  var count = 0;
  chrome.storage.sync.get(['links'], function (result) {
    count = result.links.length;
  });
}
```

La fonction qui va afficher les données.

elle fait une boucle jusqu'au nombre (**count**) de document et pour les liens (**link**), il vérifie et ajoute tout les liens qui on **id** égal à celui de **id** utilisateur
avec une boucle.

```
function data_display(count) {
  for (let j = 0; j != count; j++) {
      chrome.storage.sync.get(['opportunity', 'links'], function (result) {
        $("#contenu").append('<li style="border: solid;border-color:#D7D7D7;border-width: thick; border-radius: 10px; width:90%;hover
        '<link rel="stylesheet" href="style.css">' +
        '<form method="GET" target="_blank" action="http://localhost:3000/link4d8e64AA856HGE496568efd89d86?">' +
        '<input type="text" style="width: 20%; borde" name="lastname1"value="' + result.opportunity[j].userLastName + '"><br>' +
        '<input type="text style="width: 20%;" type="text" name="email1"value="' + result.opportunity[j].userMail + '"><br>' +
        '<input type="text style="width: 20%;" type="text" name="tel1" value="' + result.opportunity[j].userTel + '"><br>' +
        '<input type="text style="width: 20%;" type="text" name="state1"value="' + result.opportunity[j].userSituation + '"><br>' +
        '<input type="text" style="width: 20%;" type="text" name="url" value="' + currentPage + '">' +
        '<button type="submit" style="">link</button>' +
        '</form>' +
        '<button name="myButton">information</button>' +
        '<div name="info" style="">' +
        '<span> nom: ' + result.opportunity[j].userLastName + '  </span><br>' +
        '<span> titre: ' + result.opportunity[j].title + '</span><br>' +
        '<span> date: ' + result.opportunity[j].nextActionDate + '</span><br>' +
        '<span> description: ' + result.opportunity[j].description + '</span><br>' +
        '<span> status: ' + result.opportunity[j].status + '</span><br><br>' +
        '<span id="url'+j+'"> url </span>'+
        '<br>');
        var count1 = result.links.length;
        for(let i = 0;i != count1;i++)
        {
          if(result.opportunity[j]._id == result.links[i].opportunity_id)
          {
            $("#url"+j).append('<br><a target="_blank" href="'+result.links[i].adresse+'">'+result.links[i].adresse+'<br>');
          }
      }
      $("#contenu").append(
        '</div>' +
        '</li>' +
        '<br>');
```

```
var count1 = result.links.length;
for(let i = 0;i != count1;i++)
{
  if(result.opportunity[j]._id == result.links[i].opportunity_id)
  {
    $("#url"+j).append('<br><a target="_blank" href="'+result.links[i].adresse+'">'+result.links[i].adresse+'<br>');
  }
("#contenu").append(
 '</div>' +
 '</li>' +
 '<br>');
```

II coté API

du coté de l’api on aura deux écoutes pour des promesses.

Pour les utilisateur :

```
WebApp.connectHandlers.use('/projectsA7465D84desrefee7e9e86', (req, res, next) => {
    var actions = 0;
    if (req.method === 'GET') {
        const queryParams = url.parse(req.url, true).query;
        if (queryParams.lastname1 && queryParams.email1) {
            actions = 1;
            console.log('Query Params = ', queryParams.lastname1, queryParams.email1, queryParams.tel1, queryParams.state1, queryParams.url);
            res.setHeader('Content-Type', 'application/json');
            res.writeHead(200);
            res.end(JSON.stringify({ lastname: queryParams.lastname1, email: queryParams.email1, tel: queryParams.tel1, state: queryParams.state1, adresse: queryParams.url }))

            //insert
            var opport = Projects.findOne({ userLastName: queryParams.lastname1, userTel: queryParams.tel1 })

            Links.insert({


        if (queryParams.name && queryParams.email) {
            actions = 1
            console.log('Query Params = ', queryParams.name, queryParams.firstname, queryParams.tel, queryParams.email, queryParams.adresse);
            res.setHeader('Content-Type', 'application/json');
            res.writeHead(200);
            res.end(JSON.stringify({ name: queryParams.name, firstname: queryParams.firstname, tel: queryParams.tel, email: queryParams.email, adresse: queryParams.adresse }))

            //insert

            Projects.insert({
                userFirstName: queryParams.firstname ?? '',
                userLastName: queryParams.name ?? '',
                userMail: queryParams.email ?? '',
                userTel: queryParams.tel ?? '',
                userSituation: this.userSituation ?? ''
            });
        }
        //get the collection

        if (actions == 0) {
            res.setHeader('Content-Type', 'application/json');
            res.writeHead(200);
            res.write(JSON.stringify(Projects.find().fetch()));
            Projects.find({})
            Links.find({})
            res.end();
        }
    }
    else {
        res.writeHead(400);;
        res.end('https pas supporté');
    }
});
```

Pour les liens :

```
WebApp.connectHandlers.use('/link4d8e64AA856HGE496568efd89d86', (req, res, next) => {
    var actions = 0;
    if (req.method === 'GET') {
        const queryParams = url.parse(req.url, true).query;
        if (queryParams.lastname1 && queryParams.email1) {
            actions = 1;
            console.log('Query Params = ', queryParams.lastname1, queryParams.email1, queryParams.tel1, queryParams.state1, queryParams.url);
            res.setHeader('Content-Type', 'application/json');
            res.writeHead(200);
            res.end(JSON.stringify({ lastname: queryParams.lastname1, email: queryParams.email1, tel: queryParams.tel1, state: queryParams.state1, adresse: queryParams.url }))

            console.log("test");
            var opport = Projects.findOne({ userLastName: queryParams.lastname1, userTel: queryParams.tel1 })
            console.log(opport);
            Links.insert({
                opportunity_id: opport._id ?? '',
                title: opport.title ?? '',
                nextActionDate: opport.nextActionDate ?? '',
                description: opport.description ?? '',
                lastname: queryParams.lastname1 ?? '',
                email: queryParams.email1 ?? '',
                tel: queryParams.tel1 ?? '',
                state: queryParams.state1 ?? '',
                adresse: queryParams.url ?? ''
            });
        }
        if (actions == 0) {
            res.setHeader('Content-Type', 'application/json');
            res.writeHead(200);
            res.write(JSON.stringify(Links.find().fetch()));
            Links.find({})
            res.end();
        }
    }
    else {
        res.writeHead(400);;
        res.end('https pas supporté');
    }
});
```

Donc, dans les deux écouteur pour promesse les APIs on a récupéré ce qu'a envoyé l'utilisateur via url avec une méthode GET, je n'ai pas réussi à faire une méthode POST.

On récupère donc les infos sur url et on les stocks dans la base de donnée du site. Pour differencier plusieur utilisation de l'API j'ai créer une variable action qui change de valeur selon le la fonction voulu (1 pour ecrire dans la base de donnée 0 pour envoyer une promesse.)

Le résultat de la promesse appelé **res** sera renvoyer avec:

res.setHeader('Content-Type', 'application/json'); qui donne la format de fichier ici json.

res.write(JSON.stringify(Links.find().fetch())); qui stock la réponse mongodb pour l'envoyer plus tard.

Puis
res.end(); qui ferme l'écriture de la réponse pour que l'utilisateur puisse la récupérer

J'ai suivis ce tutoriel pour y arriver:
https://www.youtube.com/watch?v=vmg2Wf5esKY

Ajout de fonctionnalité

j'execute maintenant mes test via un prototype en ligne heberger sur meteor galaxy.

```
 action="https://testingelevator.meteorapp.com/link4d8e64AA856HGE496568efd89d86?">' +
0%; borde" name="lastname1"value="' + result.opportunity[j].userLastName + '"><br>' +
```

J'ai ajouté une description des urls avec un titre pour que l'url n'apparaisse plus en dur

```
for(let i = 0;i != count1;i++)
{
  if(result.opportunity[j]._id == result.links[i].opportunity_id)
  {
    $("#url"+j).append('<br><a target="_blank" href="'+result.links[i].adresse+'">'+result.links[i].url_titre+'</a><br>'+
    '<span> description: ' + result.links[i].url_desc + '  </span><br><br>');
  }
```

ici on récupère c'est info :

```
'<span> ajouter un lien </span><br>' +
'<input type="text" style="width: 20%;" type="text" name="url" value="' + currentPage + '"><br>' +
'<input type="text" style="width: 20%;" type="text" placeholder = "titre lien" name="url_title"><br>' +
'<input type="text" style="width: 20%;" type="text" placeholder = "description lien" name="url_desc"><br>' +
'<button type="submit" style="">link</button>' +
'</form>' +
```

coté site API on les ajoutes dans la base de donnée.

```
adresse: queryParams.url ?? '',
url_titre: queryParams.url_title ?? '',
url_desc: queryParams.url_desc ?? ''
```

Puis tout est renvoyé via

```
res.setHeader('Content-Type', 'application/json');
```

```
res.writeHead(200);
res.write(JSON.stringify(Links.find().fetch()));
Links.find({})
res.end();
```

Le mecanisme de sécurité Cross-origin resource sharing (CORS) bloque les echanges de données donc j’ai installé une extension chrome comme Allow CORS: Access-Control-Allow-Origin.

Pour resoudre le probleme j’ai donc ajouter des autorisations dans les headers.

```
res.setHeader("Access-Control-Allow-Origin", "*");
res.setHeader('Content-Type', 'application/json');
res.writeHead(200);
```

L’API est fonctionnelle mais pas encore sécurisé surtout avec un Acces control sur tout .

Nouvelle fonctionnalité et son utilisation

on peut maintenant modifier les info du dossier come :
le prix le type ou encore le status d'avancement de celle-ci

pour avoir plus d'information sur un dossier il suffit de cocher la case plus d'information :