

programmation web avancée

Réalisation d'une application web



Réalisé par:

- **BENOUDJIT Hadj Tahar**
- **FOTO Dimitri dylane**

Objectif de projet:

Réalisation d'une application web avec:

- Construction d'une interface web avec HTML, CSS and Vue.js
- Construction d'un server web avec Un Node.js and Express
- Authentication d'utilisateur
- les interactions CRUD avec le serveur
- Deployment sur Glitch

introduction:

dans ce projet, nous sommes sur le point de faire une boutique en ligne pour la pizza, il sert à montrer une page de bienvenue, et un bouton à la page menu où nous pouvons trouver nos produits (modèles De pizza), et une carte d'achat qui font la somme des commandes; après la validation de la commande, nous nous retrouvons dans la dernière page que nous pourrions revenir à la page principale par un button.

pour le back-end nous construisons un serveur web en utilisant Node.js qui permet à l'interface web de communiquer avec le serveur tels que nous ajoutant un nom d'utilisateur et mot de passe et pour l'authentification aussi , pour le front-end nous avons utilisé HTML,CSS,Vue.js , qui nous permet d'organiser les composants et la façon dont il ressemble , avec l'option d'ajoute et de supprimer (panier d'achat) et un calcul automatique (Total).

Lien sur glitch: <https://taharbenoudjit-firts-app-4.glitch.me/?#/>

Nom d'utilisateur: admin

mot de passe: admin

Etapas de conception:

Front-end:

1- construction de navBar on ajoutant le nom de la boutique comme un bouton et le menu bouton qui nous passer à la page de menu , bouton de connexion pour nous lier à la page d'authentification , sign in pour la page d'inscription , avec une icône pour le log out (qui n'a aucune utilité dans notre cas).

2- construire le Footer en créant une carte , et nous avons donné un titre pour la carte (ABOUT US), et en dessous trois icônes avec un espace créé par la commande <v-devider> un pour localisation de la boutique (il nous passer à «google maps»), l'autre à Gmail pour nous contacter par e-mail l'autre pour l'appel par téléphone

ps: l'idée était d'ajouter des icônes des réseaux sociaux (Facebook, Instagram,..) mais nous n'avons pas pu trouver comment le faire.

3- appelant le footer et navBar dans app.vue pour les montrer dans toutes nos pages . nous avons fait un routeur (router.js) qui nous permet de basculer entre les pages et nous l'avons utilisé dans app.vue.

4- fond pour la page principale qui est une image, avec une carte dans le côté droit pour accueillir le client et des mots de motivation et il a un bouton pour aller à la page de menu (Let's ordre).

5- la page de menu, dans le haut nous trouvons un carrousel qui nous permet de changer à trois images défèrent pour la publicité des produits ,nous avons ajoutés une image pour l'arrière-plan après une carte deviser en 2 côtés; dans le côté droit, nous avons fait une liste dans chaque ligne a un type de pizza définie par son titre en en dessous on trouve les produits avec une ligne pour diviser le titre des produits , dans la ligne un 'container' pour contenir les produits qui déplie par des cartes dans chacun un titre qui se réfèrent au nom du produit et une image en dessous une description (pour les ingrédients de produit) , et un bouton dans en dessous pour choisir le produit et l'ajouter à la facture, cette opération est pour un produit , nous avons utilisé un v-for pour faire la même chose pour les autres produits aussi et se répétera pour chaque produit dans le menu, dans le côté gauche nous trouvons la facture qui Apparaît le nom du produit et son prix en dessous le total avec la possibilité d'enlever un produit, avec le bouton de la facture nous n'aimons pas , le bouton 'valider' pour valider la commande et il nous passer à la dernière page .

6- les pages de 'sign in' et 'login' n'a aucune utilité pour le front-end, il fonctionne avec le back-end.

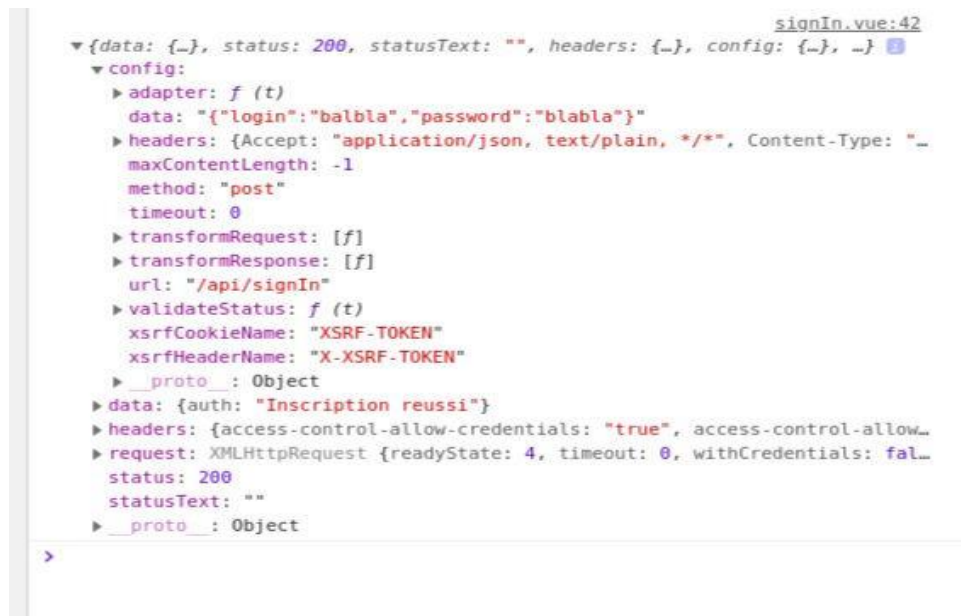
Back-end: et nous pouvons le trouver dans le fichier (server.js) de notre dossier de projet

1- a première interaction avec le serveur est de faire un test.

2- nous avons créé une authentification utilisateur qui est «admin» pour le mot de passe et le nom d'utilisateur.

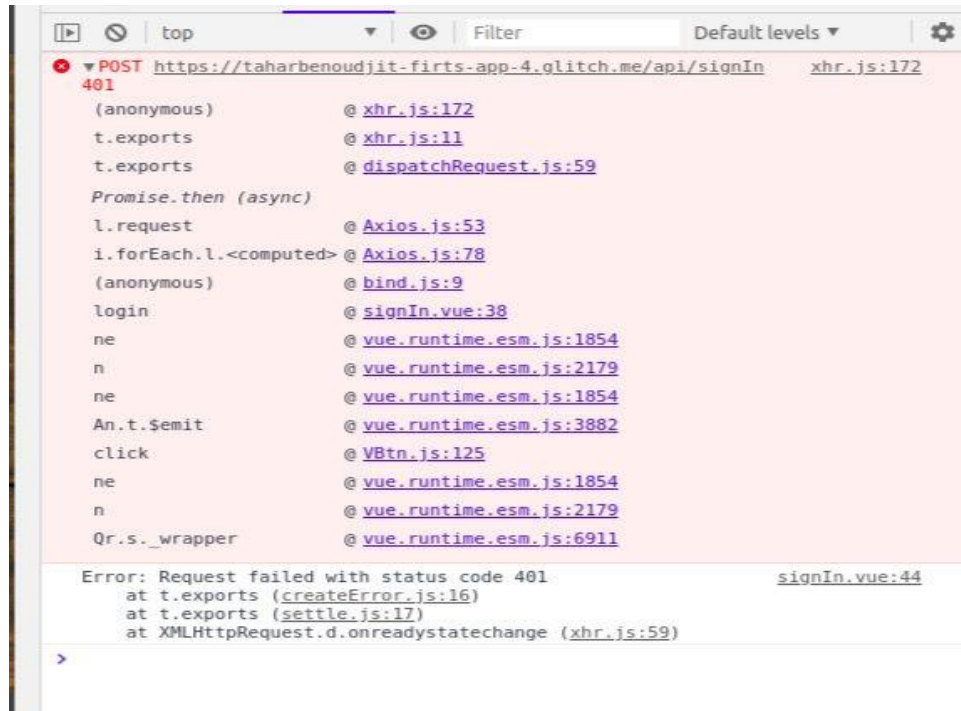
3- nous avons fait une interactions de type https.

4- pour la partie sign in si nous utilisons admin pour le nom d'utilisateur et mot de passe; la console afficher un message 401 qui n'est pas autorisé parce que c'est des informations qui sont dans la base de données déjà , sinon si nous mettons des informations d'authentification, a part admin et admin un message 200 apparaissent ce qui signifie que l'inscription a réussi et nous pouvons trouver ces informations dans la console.



```
signIn.vue:42
▼ {data: {_,}, status: 200, statusText: "", headers: {_,}, config: {_, _}}
  ▼ config:
    ▶ adapter: f (t)
      data: "{"login":"balbla","password":"blabla"}"
    ▶ headers: {Accept: "application/json, text/plain, */*", Content-Type: "...",
      maxLength: -1
      method: "post"
      timeout: 0
    ▶ transformRequest: [f]
    ▶ transformResponse: [f]
      url: "/api/signIn"
    ▶ validateStatus: f (t)
      xsrfCookieName: "XSRF-TOKEN"
      xsrfHeaderName: "X-XSRF-TOKEN"
    ▶ __proto__: Object
  ▶ data: {auth: "Inscription reussi"}
  ▶ headers: {access-control-allow-credentials: "true", access-control-allow...
  ▶ request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: fal...
    status: 200
    statusText: ""
  ▶ __proto__: Object
>
```

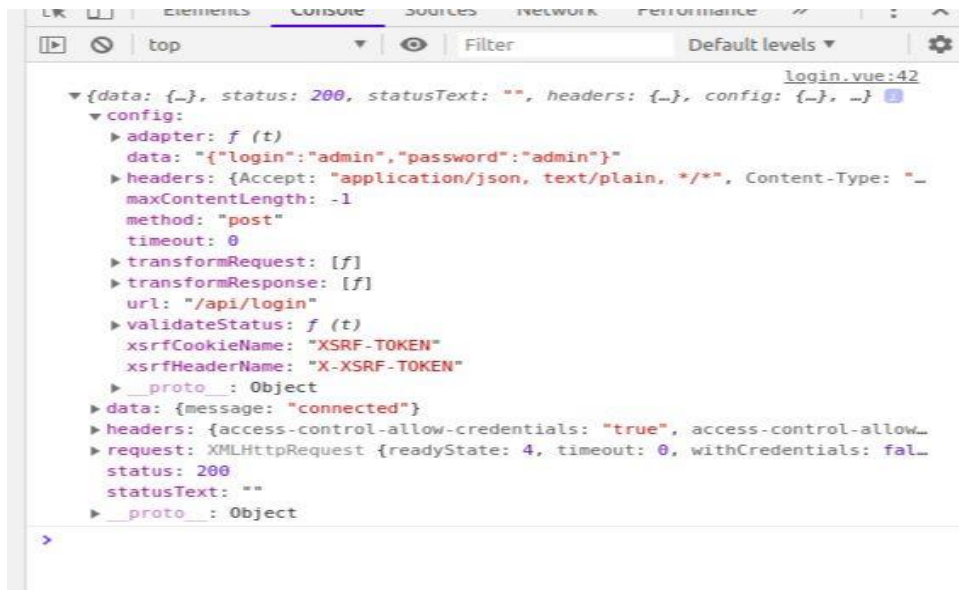
Avec des informations aléatoire



Avec admin

5- pour le login partie nous avons utilisé des informations statique pour l'authentification qui est «admin» pour le nom d'utilisateur et mot de passe si nous l'entrons correctement la console

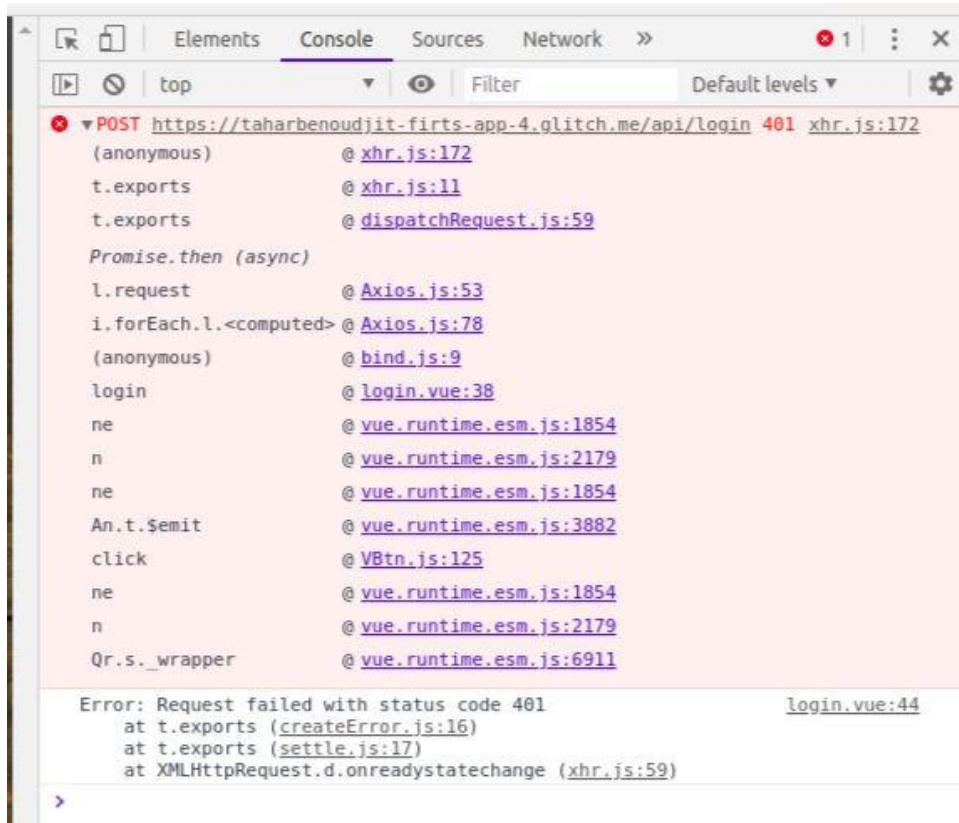
montre un message 200 qui est autorisé, si nous utilisons un autre mot de passe ou nom d'utilisateur de la console montre un message 401 ce qui signifie non-authoriser



The screenshot shows the Chrome DevTools console with the 'Console' tab selected. A log entry from `login.vue:42` displays a Vue.js `axios` request configuration. The configuration includes a POST request to `/api/login` with a data object containing `login: 'admin'` and `password: 'admin'`. The request headers include `Accept: application/json, text/plain, */*` and `Content-Type: application/json`. The response status is 200, and the message is 'connected'.

```
{data: {_, status: 200, statusText: "", headers: {_, config: {_, _}}, config: {_, _}}
  config:
    adapter: f (t)
    data: {"login": "admin", "password": "admin"}
    headers: {Accept: "application/json, text/plain, /*", Content-Type: "..."}
    maxLength: -1
    method: "post"
    timeout: 0
    transformRequest: [f]
    transformResponse: [f]
    url: "/api/login"
    validateStatus: f (t)
    xsrfCookieName: "XSRF-TOKEN"
    xsrfHeaderName: "X-XSRF-TOKEN"
    _proto_: Object
  data: {message: "connected"}
  headers: {access-control-allow-credentials: "true", access-control-allow-origins: "..."}
  request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false}
  status: 200
  statusText: ""
  _proto_: Object
}
```

Avec admin



The screenshot shows the Chrome DevTools console with the 'Console' tab selected. A log entry from `xhr.js:172` displays a failed XMLHttpRequest. The status is 401 (Unauthorized). The error message at the bottom states: 'Error: Request failed with status code 401'. The stack trace includes `login.vue:44`, `createError.js:16`, `settle.js:17`, and `xhr.js:59`.

```
POST https://taharbenoudjit-firts-app-4.glitch.me/api/login 401 xhr.js:172
(anonymous) @ xhr.js:172
t.exports @ xhr.js:11
t.exports @ dispatchRequest.js:59
Promise.then (async)
l.request @ Axios.js:53
i.forEach.l.<computed> @ Axios.js:78
(anonymous) @ bind.js:9
login @ login.vue:38
ne @ vue.runtime.esm.js:1854
n @ vue.runtime.esm.js:2179
ne @ vue.runtime.esm.js:1854
An.t.$emit @ vue.runtime.esm.js:3882
click @ VBtn.js:125
ne @ vue.runtime.esm.js:1854
n @ vue.runtime.esm.js:2179
Qr.s._wrapper @ vue.runtime.esm.js:6911
Error: Request failed with status code 401 login.vue:44
  at t.exports (createError.js:16)
  at t.exports (settle.js:17)
  at XMLHttpRequest.d.onreadystatechange (xhr.js:59)
```

Avec des informations aléatoire

problèmes: nous ne pouvons pas faire les bonnes tailles pour les cartes dans les petit écrans