



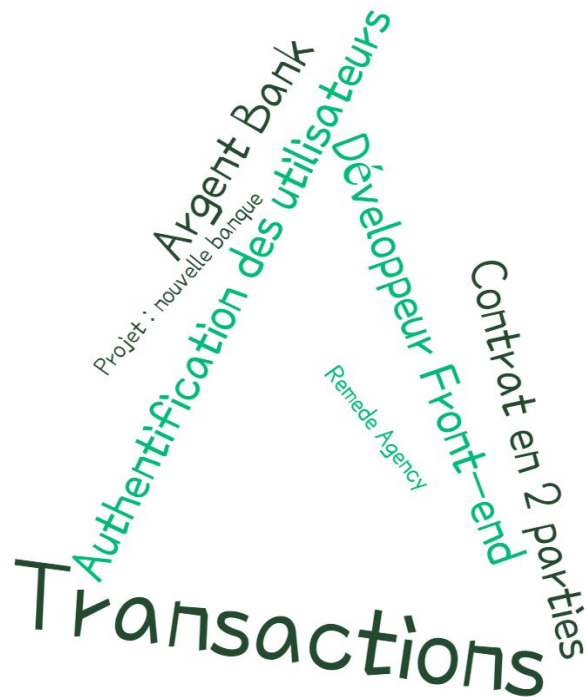
Projet

**ARGENTBANK**

Utilisez une API pour un compte  
utilisateur bancaire avec React



# Contexte



# Adaptation projet

Pour s'imprégner du projet, documents et outils à disposition :

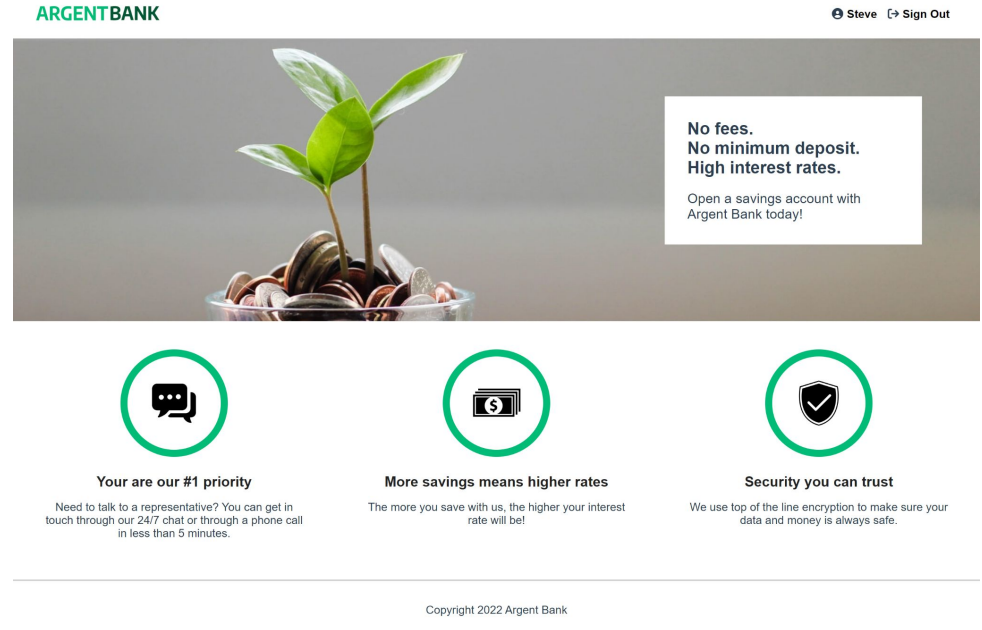
- Mail d'informations et débrief sur le projet
- Html statique / Css / Wireframes :  
<https://github.com/OpenClassrooms-Student-Center/Project-10-Bank-API/tree/master/designs>
- Issues :  
[https://github.com/OpenClassrooms-Student-Center/Project-10-Bank-API/tree/master/github/ISSUE\\_TEMPLATE](https://github.com/OpenClassrooms-Student-Center/Project-10-Bank-API/tree/master/github/ISSUE_TEMPLATE)
- Repo existant : <https://github.com/OpenClassrooms-Student-Center/Project-10-Bank-API>
- Base de données : sur le serveur via : `populate-db` ou MongoDB Compass

Projet codé

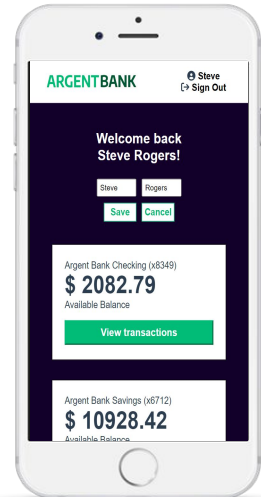
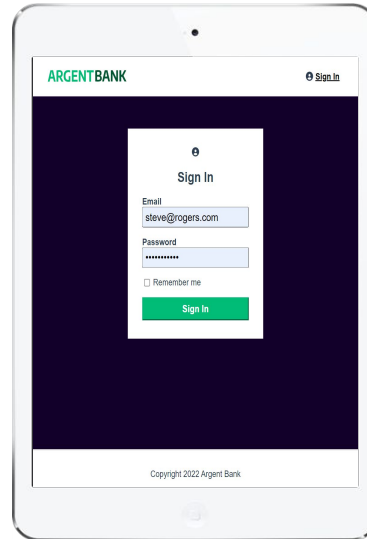
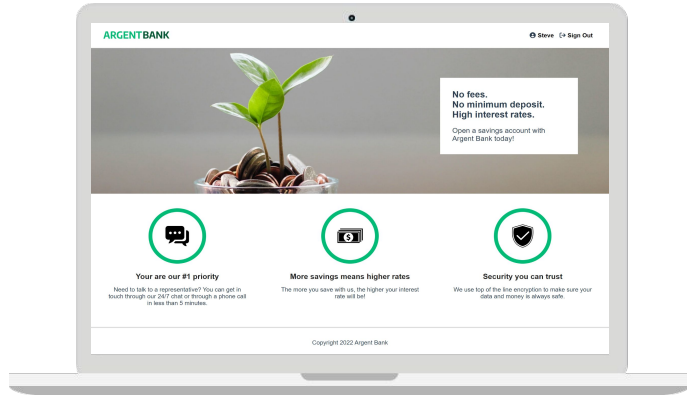
# Projet codé

- Projet codé au HTML / CSS / Wireframes

- Comprend :
  - accueil
  - login
  - profil
  - transactions

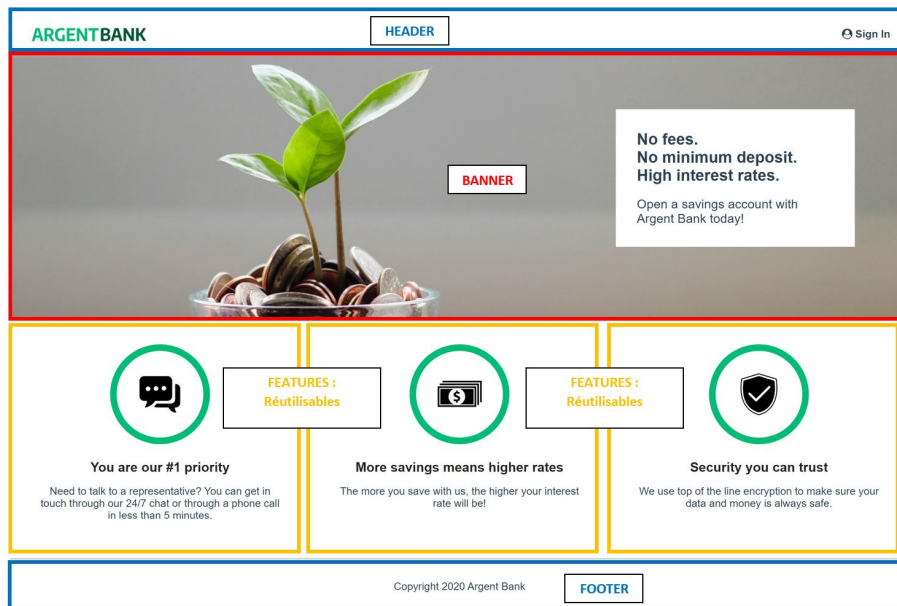
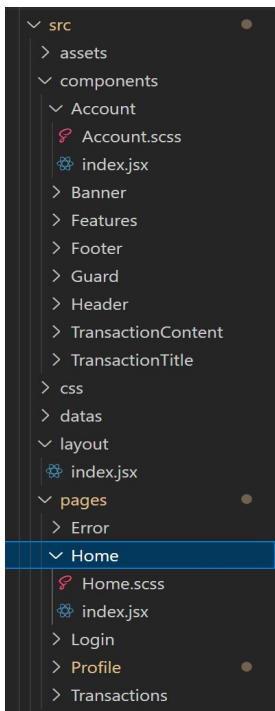


# Responsive



# Composants React

- Doc Word Décomposition

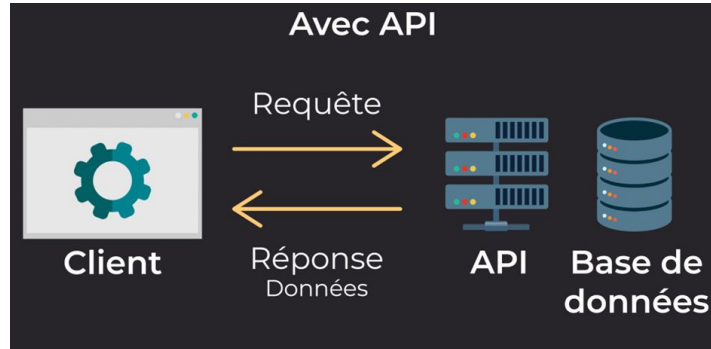


API



# Rappel API

- Le client va demander à l'API une information, celle-ci va aller chercher cette information dans la BDD puis la renvoyer au client dans un second temps



- Service web RESTfull : utilise des méthodes HTTP pour récupérer et publier des données dans un format JSON, entre client et un serveur

# Call API : Axios

- Faire des **requêtes HTTP vers des ressources externes**
- Axios : <https://github.com/axios/axios>
- Spécifier les paramètres de configuration par défaut qui seront appliqués à chaque demande

```
axios.defaults.baseURL = 'http://localhost:3001/api/v1';

//instance replace the value default for the token
export const instance = axios.create({
  headers: {
    common: {
      Authorization: 'AUTH_TOKEN_FROM_INSTANCE',
    },
  },
});

export const api = {
  /**
   * Get the profile of the corresponding token
   * @returns
   */
  getProfile: () => {
    return instance
      .post('/user/profile')
      .then((response) => response.data);
  },
};
```

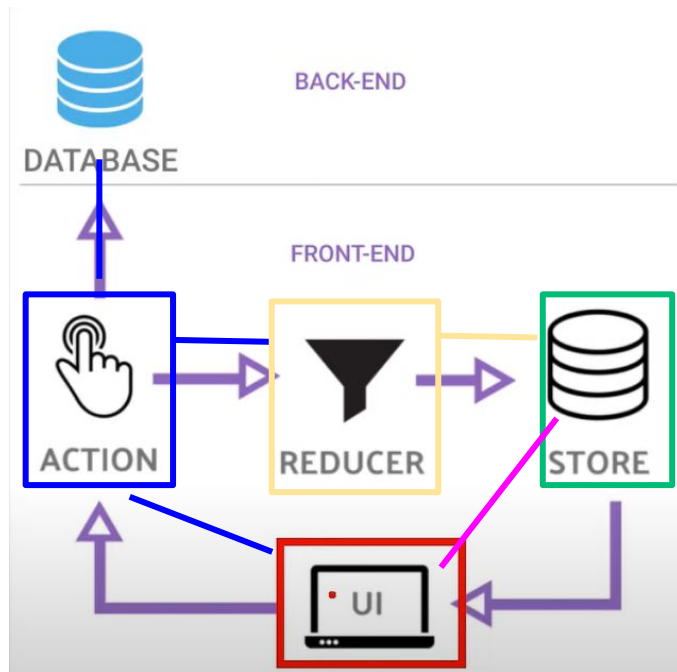
# Redux

# Redux

- Redux reprend la notion de **state** :
  - Il détermine comment le composant doit se comporter ou être rendu
- Redux est une **bibliothèque indépendante** qui nous aide à gérer notre état (state) en donnant à nos composants l'état dont il a besoin via un store central.
- Redux **stocke tout l'état de notre application** dans une arborescence d'objets, qui ne change pas.

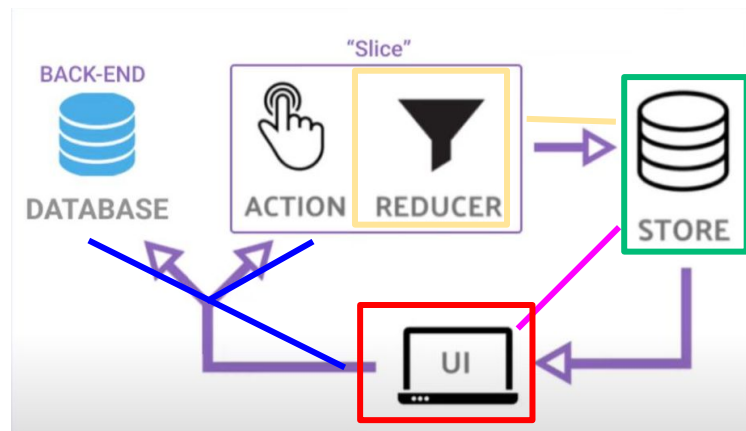
# Comprendre Redux

- UI : exemple => un user veut supprimer une image
- **Action** se déclenche : **dispatch**
  - l'action dira à la bdd de suppr. l'image
  - l'action dira au reducer de suppr. l'image du store
- **Réducer** est le seul élément qui peut changer les éléments dans le store
- UI puise les éléments dans le store
  - Pas besoin d'interroger la bdd en permanence
  - Tous les éléments qui puisent dans le store se modifieront, si une donnée évolue.



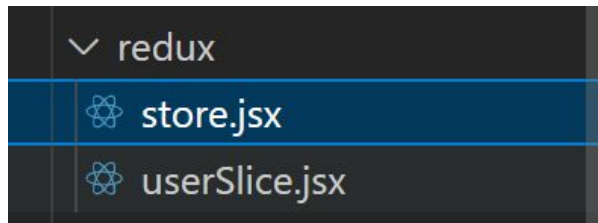
# Redux Toolkit

- Redux-Toolkit est **un outil** qui permet de faciliter l'utilisation de Redux.
- L'interface UI récupère toujours les données mise à jour à partir du store ( en rose )
  - **Exemple : un user clique sur un élément qu'il veut supprimer**
- **L'action est "fusionnée" avec le reducer** => slice
- On peut toujours incrémenter la base de donnée
- **L'action va agir avec le reducer :**
  - on va demander telle action
  - au reducer pour que le store s'actualise



# Redux Toolkit dans VSCode

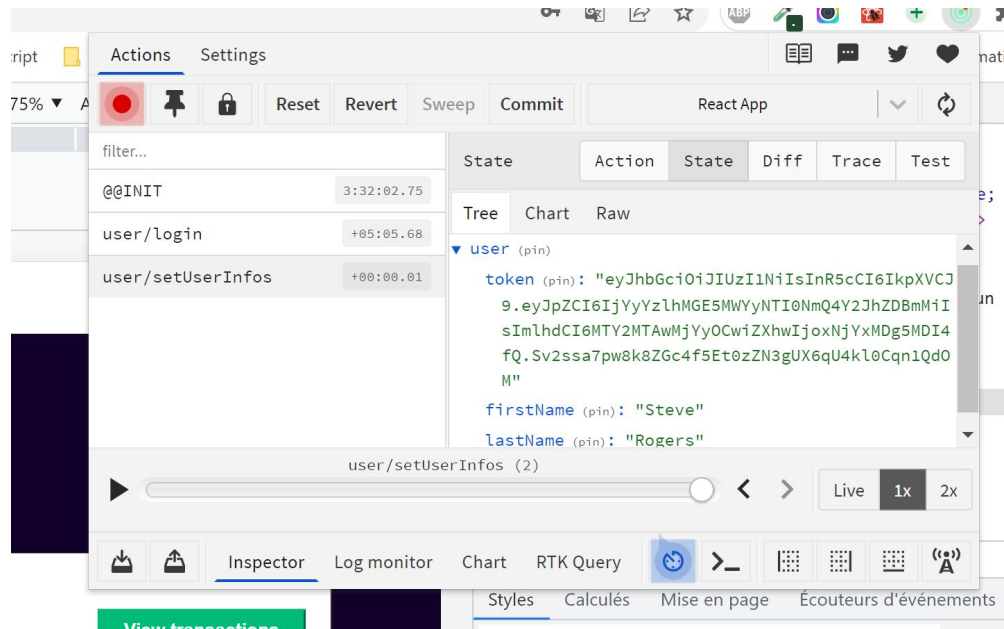
- Explication du process pour utiliser Redux ToolKit et connecter le store.
  - Voir process à suivre dans toolkit.text



```
store.jsx  X
src > redux > store.jsx > ...
1  import { configureStore } from '@reduxjs/toolkit';
2  import { userReducer } from './userSlice';
3
4  export default configureStore({
5    reducer: {
6      user: userReducer,
7    },
8  });
9
```

# Redux DevTools

- Extension Google Chrome
- Vérifier la connexion au Store
- Les reducers
- Enregistrement des étapes





# Phase 1 : Authentication

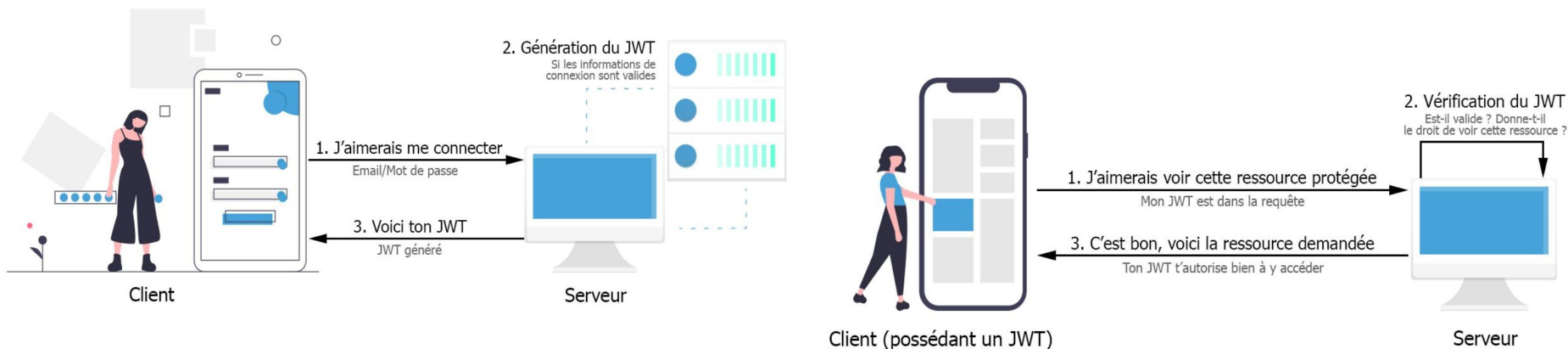
# Phase 1 du projet

- L'utilisateur peut :
  - visiter la page d'accueil.
  - se connecter au système.
  - accéder à ses informations bancaires
  - se déconnecter du système.
- L'utilisateur ne peut voir les informations relatives à son propre profil qu'après s'être connecté **avec succès**.
- L'utilisateur peut modifier le profil et conserver les données dans la base de données.

# Authentification utilisateur

- Les **tokens d'authentification** permettent aux utilisateurs de se connecter une seule fois à leur compte.

Au moment de se connecter, ils recevront leur *token* et le renverront automatiquement à chaque requête par la suite. Ceci permettra au back-end de vérifier que la requête est authentifiée.



# Authentification utilisateur

- L'onglet « **Réseau** » de Chrome DevTools :
  - vérifier qu'une fois l'utilisateur connecté, chaque requête provenant du front-end contient bien un en-tête « Authorization »

The image shows a web application interface for 'ARGENTBANK' and the Chrome DevTools Network tab. The web application displays a welcome message for 'Steve Rogers' and two account balances: 'Argent Bank Checking (x6349)' with a balance of \$2082.79 and 'Argent Bank Savings (x6712)' with a balance of \$10928.42. The DevTools Network tab shows a list of requests, with the 'profile' request selected. The 'En-têtes de réponse' (Response Headers) section is expanded, showing the 'Access-Control-Allow-Headers' header set to 'authorization'.

ARGENTBANK

Steve Sign Out

Welcome back Steve Rogers !

Edit Name

Argent Bank Checking (x6349)

\$ 2082.79

Available Balance

View transactions

Argent Bank Savings (x6712)

\$ 10928.42

Available Balance

View transactions

Tout Fetch/XHR JS CSS Image Multimédia Police Document WS Wasm Fichier manifeste Autre

☐ A bloqué les cookies ☐ Requêtes bloquées ☐ Requêtes tierces

2000 ms 4000 ms 6000 ms 8000 ms 10000 ms 12000 ms 14000 ms 16000 ms 18000 ms

Nom X En-têtes Aperçu Réponse Initiateur Délai

login

bundle.js

axe.js

argentBankLo...

react\_devtools...

highlighter.js

ws

manifest.json

favicon.ico

logo192.png

login

login

profile

profile

▼ Général

URL de requête: http://localhost:3001/api/v1/user/profile

Mode de requête: OPTIONS

Code d'état: 204 No Content

Adresse distante: [::1]:3001

Règlement sur les URL de provenance: strict-origin-when-cross-origin

▼ En-têtes de réponse Afficher la source

Access-Control-Allow-Headers: authorization

Access-Control-Allow-Methods: GET, HEAD, PUT, PATCH, POST, DELETE

Access-Control-Allow-Origin: \*

Connection: keep-alive

Content-Length: 0

Date: Sun, 21 Aug 2022 13:46:56 GMT

Keep-Alive: timeout=5

Vary: Access-Control-Request-Headers

X-Powered-By: Express

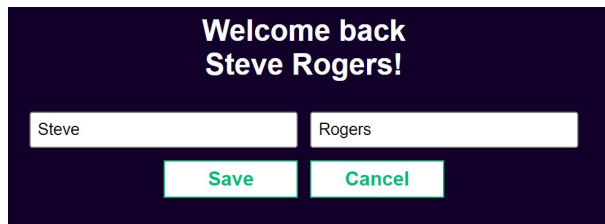
# Token et Redux

- **Axios** permet de créer des **instances par défaut personnalisées**
  - <https://github.com/axios/axios#custom-instance-defaults>
- Ajout des reducers login et logout dans le slice
  - Appeler l'instance Axios pour **l'authentification token**
- Sur la page Login : appeler **l'action** pour récupérer le token
- Dans le composant Header : utiliser la donnée token
- Utilisation du composant **Guard** :
  - permet de protéger la page (ou route) souhaitée par des autorisations
- Vérification si **le store récupère bien la clé Token** quand on se connecte (devTools)

Mise à jour du profil

# Mise à jour du profil

- L'utilisateur a la possibilité de **modifier son nom et prénom** dans le profil



Welcome back  
Steve Rogers!

Steve Rogers

Save Cancel

- Utiliser `UseSelector()` pour **récupérer nom et prénom via le réducer**
- Utiliser l'api pour la **mise à jour de ces données** en récupérant un state, pour la **sauvegarde** des informations du profil ( fonction `save()`)

## Phase 2 : Transactions



# Phase 2 du projet

- Visualiser toutes leurs transactions pour le mois en cours.
- Afficher les détails d'une transaction dans une autre vue.
- Ajouter, modifier ou supprimer des informations sur les transactions.

dropdown pour  
chaque transaction

ARGENTBANK

Steve → Sign Out

Argent Bank Checking (x8349)

**\$2,082.79**  
Available Balance

	DATE	DESCRIPTION	AMOUNT	BALANCE
^	June 20th, 2020	Golden Sun Bakery	\$5.00	\$2082.79
		Transaction Type: Electronic		
		Category: Food		
		Notes:		
▼	June 20th, 2020	Golden Sun Bakery	\$10.00	\$2087.79
▼	June 20th, 2020	Golden Sun Bakery	\$20.00	\$2097.79
▼	June 20th, 2020	Golden Sun Bakery	\$40.00	\$2147.79
▼	June 20th, 2020	Golden Sun Bakery	\$50.00	\$2187.79

SWAGGER

# Swagger

- Swagger est un langage de description d'interface permettant de décrire des API RESTful exprimées à l'aide de JSON.
- Swagger est utilisé avec toute une série d'outils logiciels open source pour concevoir, créer, documenter et utiliser des services Web RESTful.
- Utilisé en Back-end, très utile en Front-end pour connaître les éléments clés sur chaque endpoint défini.

# Swagger

- <https://editor.swagger.io/>

The image shows the Swagger Editor interface. On the left, a code editor displays a YAML API definition for a 'Bank Argent API'. The definition includes metadata like title, version, and host, and two endpoints: a POST endpoint for '/user/login' and a POST endpoint for '/user/signup'. The right panel provides a visual representation of the API, organized into modules. The 'User Module' contains three endpoints: POST /user/login (Login), POST /user/signup (Signup), and POST /user/profile (User Profile API). The 'Transaction Module' contains three endpoints: GET /user/{accountId}/transactions (fetch all transactions of a user for current month), GET /user/{accountId}/transactions/{transactionId} (Find transaction by id), and PUT /user/{accountId}/transactions/{transactionId} (modify a transaction). Each endpoint is represented by a colored box with its method, path, and a brief description.

```
1 swagger: "2.0"
2 info:
3   title: Bank Argent API documentation
4   description: Contains all available API endpoints in this codebase
5   version: "2.0.0"
6   termsOfService: "http://swagger.io/terms/"
7   host: localhost:3001
8   basePath: /api/v2
9   schemes:
10   - http
11 paths:
12   /user/login:
13     post:
14       tags:
15         - User Module
16       summary: Login
17       description: API for Login
18       parameters:
19         - in: body
20           name: body
21           description: Login Payload
22           required: true
23           schema:
24             $ref: "#/definitions/Login"
25       produces:
26         - application/json
27       responses:
28         "200":
29           description: Login Successfully
30           schema:
31             $ref: "#/definitions/LoginResponse"
32         "400":
33           description: Invalid Fields
34         "500":
35           description: Internal Server Error
36   /user/signup:
37     post:
38       tags:
39         - User Module
40       summary: Signup
41       description: API for Signup
42       parameters:
43         - in: body
```

### User Module

- POST /user/login Login
- POST /user/signup Signup
- POST /user/profile User Profile API
- PUT /user/profile User Profile API

### Transaction Module

- GET /user/{accountId}/transactions fetch all transactions of a user for current month
- GET /user/{accountId}/transactions/{transactionId} Find transaction by id
- PUT /user/{accountId}/transactions/{transactionId} modify a transaction

# Swagger : transactions

- API de transaction proposées
- Les éléments clés à spécifier pour chaque point de terminaison de l'API sont les suivants :
  - La méthode HTTP (par exemple, GET, POST, etc.)
  - La route (par exemple, /store/inventory)
  - La description de ce à quoi correspond le point de terminaison.
  - Les paramètres possibles pour tenir compte des différents scénarios.
  - Les différentes réponses avec les codes de réponse correspondant.

JS Doc

# JSDoc

- JSDoc est un langage de balisage utilisé pour documenter les codes sources Javascript
- <https://jsdoc.app/about-getting-started.html>
- Projet : fonctions et méthodes documentées

```
/**
 * Update the profile of the corresponding token
 * @param {string} firstName
 * @param {string} lastName
 * @returns
 */
updateProfile: (firstName, lastName) => {
  return (
    instance
      //méthode Put
      .put('/user/profile', {
        firstName,
        lastName,
      })
      .then((response) => response.data)
  );
},
```


# Repository





# Repo Github

- Lien vers le repo Github :

[https://github.com/Laurene45/LaureneCourde\\_13\\_29062022/tree/main/bankapi](https://github.com/Laurene45/LaureneCourde_13_29062022/tree/main/bankapi)

 Laurene45 mise à jour du fichier swagger c7d98c9 20 hours ago [History](#)

..		
public	1er chargement du projet React argentBank	2 months ago
src	mise à jour du css responsive	20 hours ago
swagger	mise à jour du fichier swagger	20 hours ago
.eslintrc.js	modification du paramètres install prop-type	2 months ago
.gitignore	1er chargement du projet React argentBank	2 months ago
.prettierrc	1er chargement du projet React argentBank	2 months ago
README.md	1er chargement du projet React argentBank	2 months ago
package-lock.json	ajout proptype et redux toolkit dans les dépendances	last month
package.json	ajout proptype et redux toolkit dans les dépendances	last month

 README.md 

## Argent Bank API

Use an API for a bank user account with React This codebase contains the code needed to run the backend for Argent Bank.

# Avez-vous des questions ?

