

Projet Web 4A : Life Manager

<https://backinbyte-web-project-2.glitch.me/>

Equipe :

FSI1

Axel DRAN

Sébastien BERNARD

Septembre-Octobre 2019

Table des matières

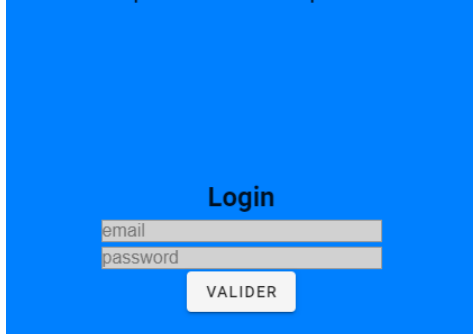
Description du projet	3
Etapes de conception, réalisation, déploiement	3
Ajouter un événement dans le calendrier	3
Supprimer un événement dans le calendrier	4
Partie Serveur	4
Affichage d'événements différents selon l'utilisateur connecté.....	4
Récupérer l'adresse mail dans la vue calendar.....	4
Modifier un événement	4
Persistance des données.....	5
Difficultés rencontrées.....	7
Problème pour mettre à jour l'événement sélectionné.....	7
Problème pour charger un tableau d'événements selon l'utilisateur connecté.....	7

Description du projet

Notre projet, Life Manager, est un planning personnel que l'utilisateur peut compléter comme il le souhaite.

Bienvenue sur Life Manager !

Le site qui va vous simplifier la vie



Pour ce faire, l'utilisateur doit d'abord se connecter à son compte avant de pouvoir accéder à son planning personnel.

Il peut ajouter, voir, mettre à jour et supprimer des événements de son planning.

Les données de notre site sont persistantes car elles sont sauvegardées dès que l'utilisateur fait la moindre action sur le site.

Identifiants par défaut :

user@email.com / password
emma@email.com / emma02
fnatic@email.com / thebest
julien@email.com / lebro

Etapas de conception, réalisation, déploiement

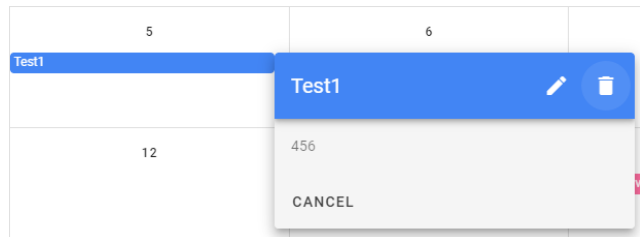
Ajouter un événement dans le calendrier

La première étape que nous souhaitons réaliser était de proposer la possibilité d'ajouter un événement dans un composant Calendrier. Pour ce faire, nous avons consulté la documentation vuetify afin de comprendre comment faire passer des données entre composants. Nous avons ainsi créé le composant Popup qui permet d'ouvrir un formulaire lors de l'appui sur un bouton. Les données collectées sont ensuite acheminées jusqu'au composant Calendrier et ajoutées à un tableau d'événements.



Supprimer un événement dans le calendrier

Pour supprimer un événement, nous avons créé une méthode dédiée appelée lors de l'appui sur un bouton. Afin de connaître l'événement à supprimer, nous nous sommes inspirés de la solution proposée dans un exemple de la documentation Vuetify du composant Calendrier. Lorsqu'un événement est sélectionné, il est temporairement gardé en mémoire dans un champ dédié. On peut ensuite appuyer sur les boutons supprimer ou modifier qui appelleront des méthodes qui s'appliqueront bien à cet événement.



Partie Serveur

Pour la partie serveur, après avoir compris comment fonctionnaient les api get et post et comment les utiliser, nous avons suivi un tutoriel pour pouvoir ajouter des identifiants et les utiliser afin de se connecter. Avec l'ajout de la vue login, nous avons une interface qui nous permet de nous connecter et qui envoie un message d'erreur si les identifiants sont erronés. Enfin, un cookie est conservé par l'utilisateur pendant 24 heures (durée pouvant être réduite), ce qui lui permet de se connecter directement sans avoir à se connecter une nouvelle fois.

Affichage d'événements différents selon l'utilisateur connecté

Après avoir mis en place un système d'authentification nous souhaitons que chaque utilisateur puisse consulter son propre calendrier. Auparavant tous les événements étaient contenus dans le même tableau et tous les utilisateurs voyaient donc les mêmes événements dans le calendrier. Pour parvenir à notre objectif nous avons ainsi créé un tableau d'événements pour chaque utilisateur. Dès qu'un utilisateur se connecte, nous sélectionnons le bon tableau à l'aide de l'adresse email de l'utilisateur.

Récupérer l'adresse mail dans la vue calendar

Afin de sélectionner le bon tableau d'événements à afficher en fonction de l'utilisateur connecté, nous avons besoin de récupérer son email ce qui va nous permettre d'afficher de manière certaine les bons événements. Avec la vue home qui ici est la vue parent, nous récupérons, grâce à l'api get login, les identifiants de connexion. Nous envoyons ensuite l'email à la vue enfant calendar qui va pouvoir afficher les bons événements.

Modifier un événement

Après avoir permis d'ajouter, lire et supprimer des événements, il nous restait à ajouter la possibilité d'en modifier afin de respecter l'acronyme CRUD.

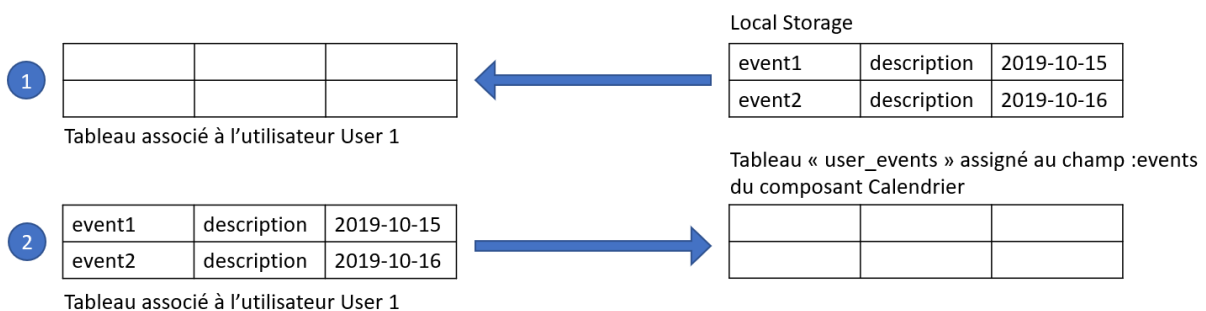
Au début, la stratégie était simple, créer un nouveau composant Popup permettant l'ouverture d'un dialogue qui cette fois-ci permettrait de modifier un événement.

Puis permettre d'ouvrir ce composant dans la v-card ouverte lors de la sélection d'un événement (la case affichée lors d'un clic sur un événement). Il faudrait alors récupérer les informations complétées par l'utilisateur et les associer à l'événement sélectionné. Après différents problèmes rencontrés (cf. « Difficultés rencontrées »), nous sauvegardons les données de l'événement sélectionné avant de les faire parvenir à la vue `popupUpdate` qui permet l'ouverture d'un dialogue afin de saisir les informations à modifier. Nous mettons ensuite à jour l'événement sélectionné en lui attribuant les informations reçues.

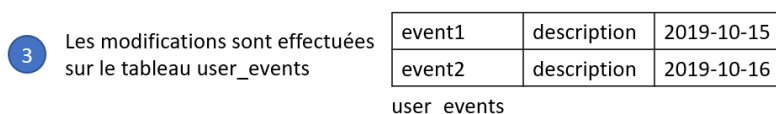
Persistence des données

Afin que les modifications effectuées sur les événements soient sauvegardées, nous avons créé une méthode qui transfère le contenu du tableau d'événements utilisé lors d'une session utilisateur (le tableau « `user_events` ») vers le tableau d'événements associé à cet utilisateur. Nous appelons cette méthode chaque fois qu'une opération est réalisée sur les événements (création, mise à jour ou suppression). De plus, cette méthode sauvegarde ensuite le tableau associé à l'utilisateur connecté dans le stockage local du navigateur (`localStorage`). Lors du chargement du composant Calendrier, les données stockées dans `localStorage` sont transférées dans les tableaux d'événements des utilisateurs. Ainsi un utilisateur qui se déconnecte puis revient sur le site retrouvera les dernières modifications effectuées.

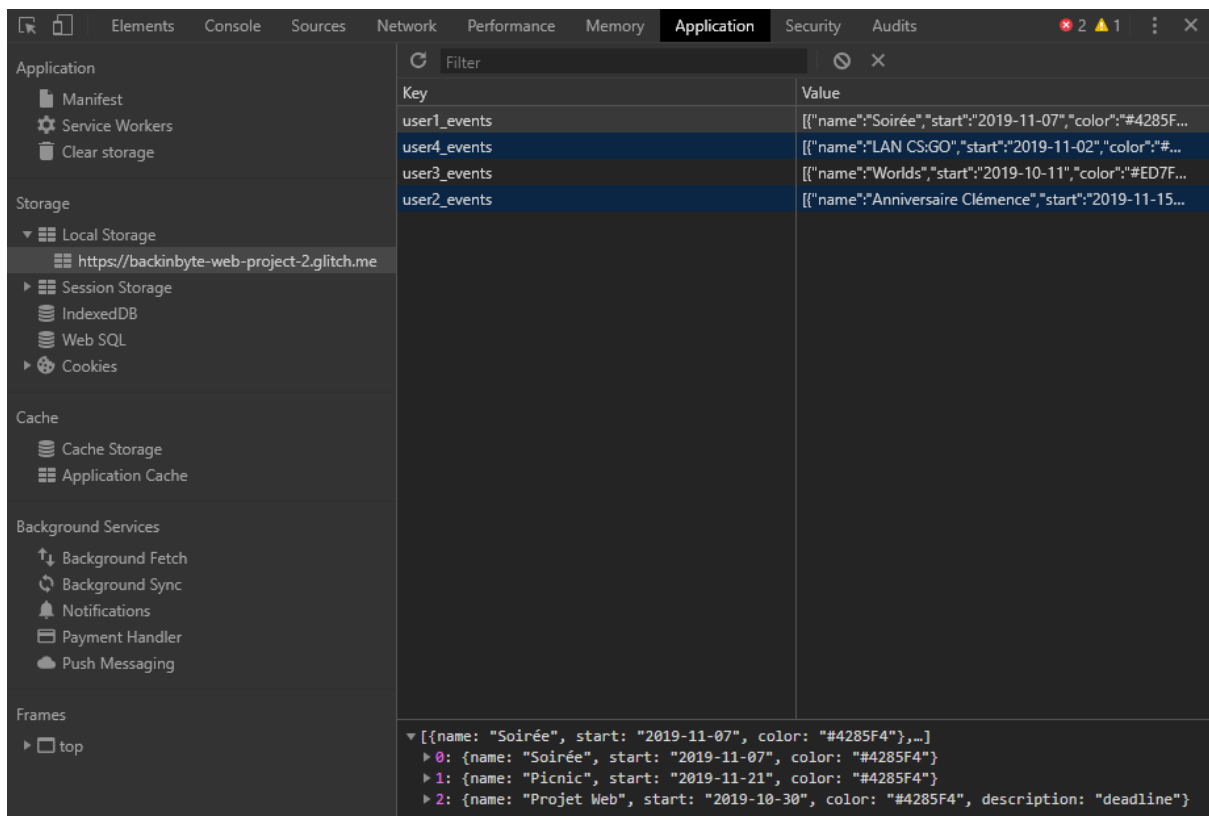
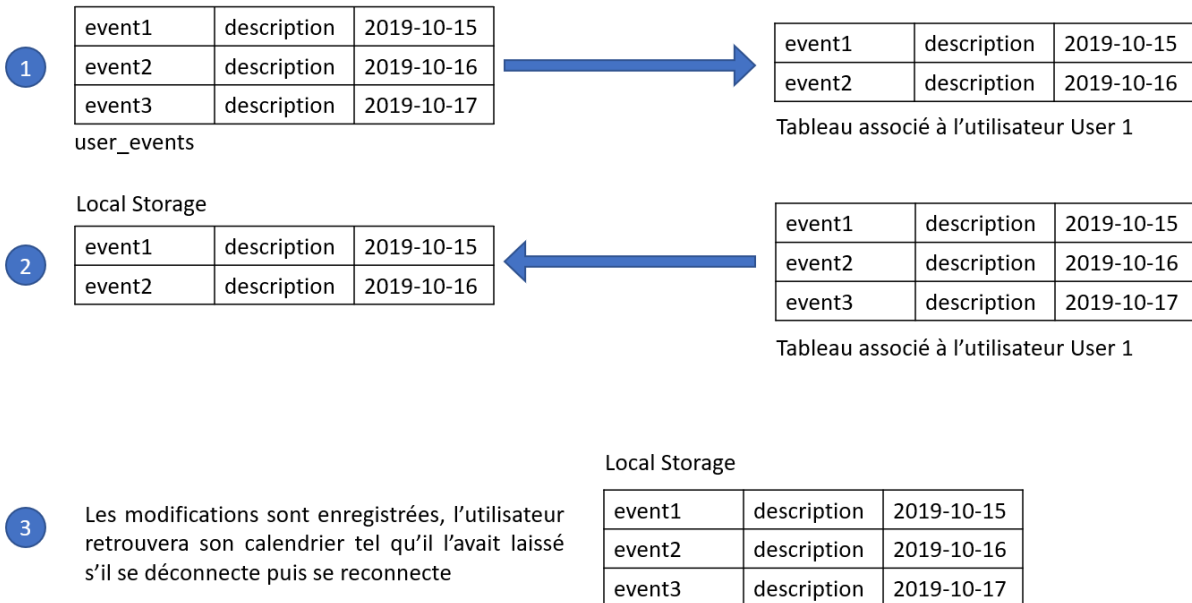
Chargement du composant Calendrier



Modifications effectuées pendant la session



Exemple : ajout d'un événement



Onglet Local Storage du navigateur où sont stockées les données

Difficultés rencontrées

Problème pour mettre à jour l'événement sélectionné

Malheureusement, la propriété `this.selectedEvent` qui est censée contenir l'événement sélectionné ne présentait aucune valeur après que l'on ait passé les données de l'événement dans la vue `popupUpdate`, laquelle ouvre un dialogue et permet de modifier les champs souhaités d'un événement. Sans `selectedEvent`, il fallait donc sauvegarder l'événement d'une autre façon avant de faire passer ses données vers cette vue. Mais après avoir testé diverses façons de lancer une fonction permettant cela, nous n'avons pas trouvé, dans un premier temps, d'autres moyens que de cliquer sur un premier bouton permettant de sauvegarder l'événement sélectionné, avant de proposer un second bouton pour lancer l'ouverture du dialogue de `popupUpdate`.

Nous avons finalement trouvé une solution pour éviter de devoir utiliser deux boutons en appelant notre fonction de sauvegarde juste après la ligne assignant l'événement sélectionné à `selectedEvent`.

Problème pour charger un tableau d'événements selon l'utilisateur connecté

Afin d'afficher les bons événements selon l'utilisateur connecté nous avons dans un premier temps tenté d'assigner au champ `:events` du composant `calendrier` le tableau d'événements correspondant en utilisant `$refs`. Cela était cependant sources de problèmes et nous avons finalement opté pour la solution consistant à assigner un tableau vide au champ `:events`, et de remplir ce tableau avec les événements du tableau correspondant à l'utilisateur se connectant.