Module de messagerie

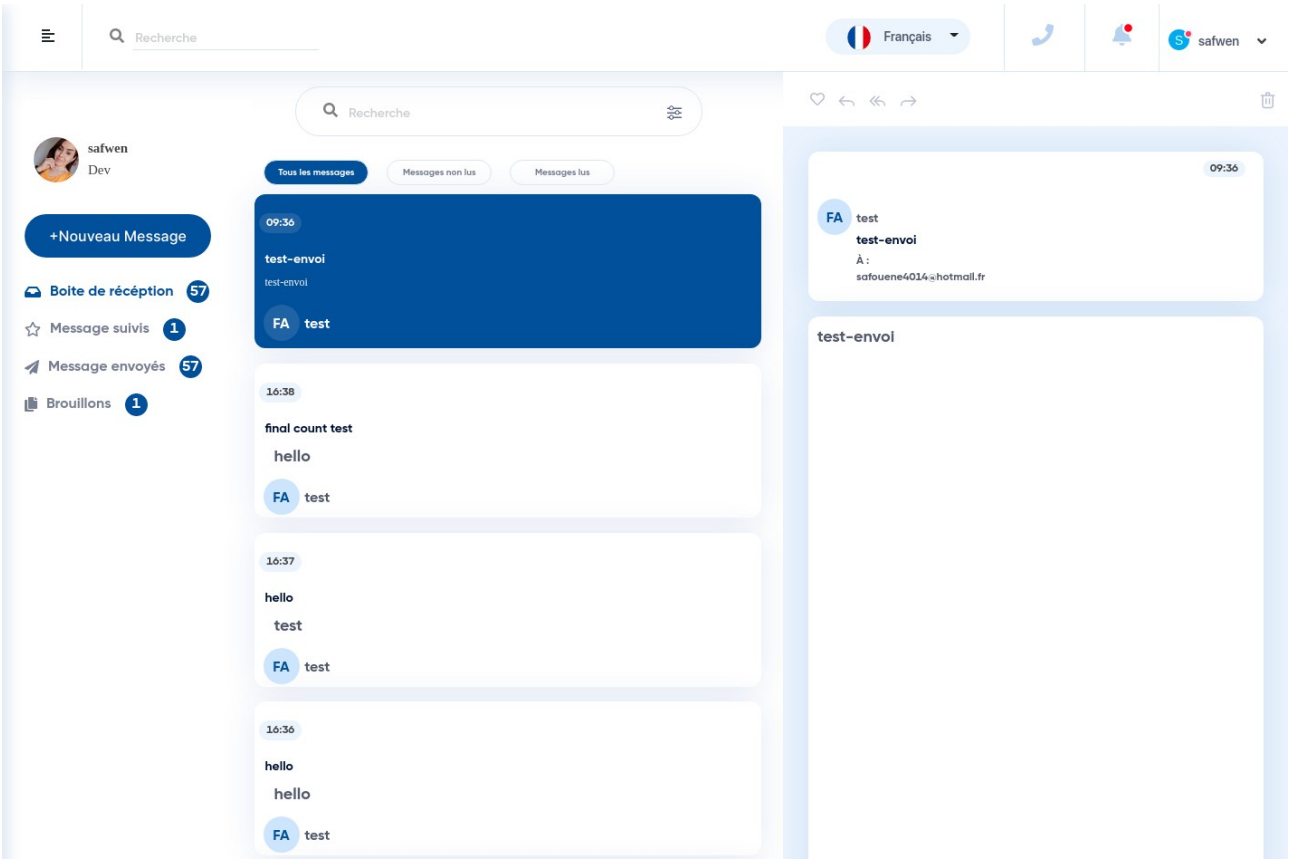
1. Introduction

L'interface de messagerie affiche la boite de réception interne de messagerie ainsi que les message envoyés, bruillons et les messages declaré important, aussi permet d'envoyer un message à un autre utilisateur de l'application.

2. Architecture de module

le module de messagerie est partager sur plusieurs composant comme illustrer ci-dessous.

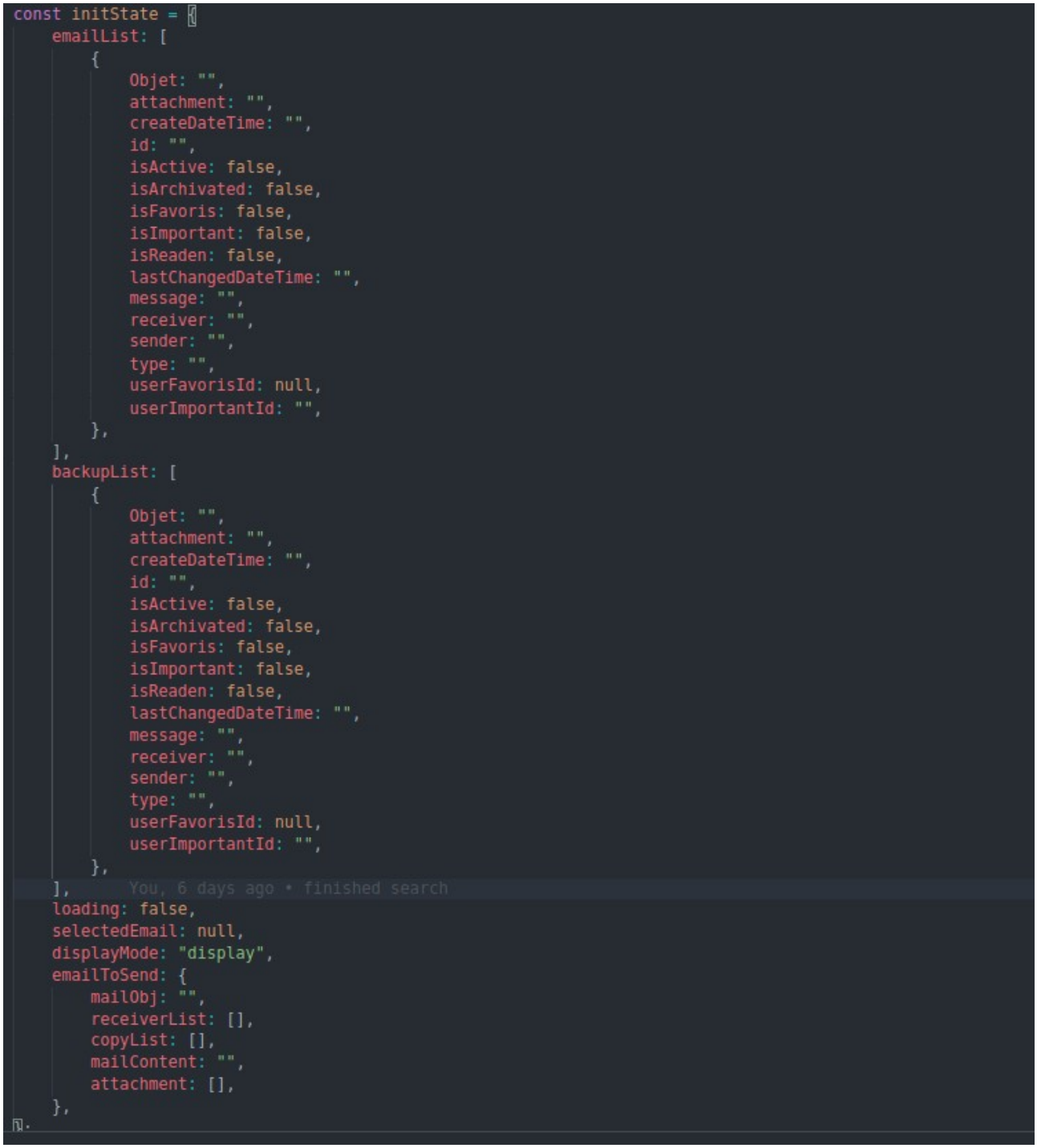
composant	Role
MailRoute	Le composant principale de module
Received	Le composant qui gere l'affichage de la liste de message (prend en paramètre liste des messages)
EmailDisplay	Le composant qui gère l'affichage d'un message selectioné
NewEmail	Le composant qui permet d'ecrire un message
EmptyEmail	Ce composant s'affiche si aucun message est séléctioner
SendEmailHeader	Sera afficher quand un utilisateur veut envoyer un message
EmailHeader	Sera afficher quand un email est selectioner



3.Gestion des fonctionalités

Puisque le module est partager sur plusieurs composants , on doit trouver une façon de partager le contenu entre ses deffirents composant et modifier l'etat d'un composant en fonction d'un autre . C'est pour sa on a crée un context (de ContextAPI React) avec un Etat initiale.

Le figures ci-dissous représente les deffirents parties du context:



```
const initState = {
    emailList: [
        {
            Objet: "",
            attachment: "",
            createDateTime: "",
            id: "",
            isActive: false,
            isArchivated: false,
            isFavoris: false,
            isImportant: false,
            isReaden: false,
            lastChangedDateTime: "",
            message: "",
            receiver: "",
            sender: "",
            type: "",
            userFavorisId: null,
            userImportantId: "",
        },
    ],
    backupList: [
        {
            Objet: "",
            attachment: "",
            createDateTime: "",
            id: "",
            isActive: false,
            isArchivated: false,
            isFavoris: false,
            isImportant: false,
            isReaden: false,
            lastChangedDateTime: "",
            message: "",
            receiver: "",
            sender: "",
            type: "",
            userFavorisId: null,
            userImportantId: "",
        },
    ],      You, 6 days ago • finished search
    loading: false,
    selectedEmail: null,
    displayMode: "display",
    emailToSend: {
        mailObj: "",
        receiverList: [],
        copyList: [],
        mailContent: "",
        attachment: [],
    },
};
```

- **InitState représente l'etat initiale de context**

  - emailList, backupList représentent la liste des messages à afficher
  - loading présente une variable booléene qui permet de savoir si un appel api est en cours de chargement.
  - selectedEmail répresente l'id un message selectioner

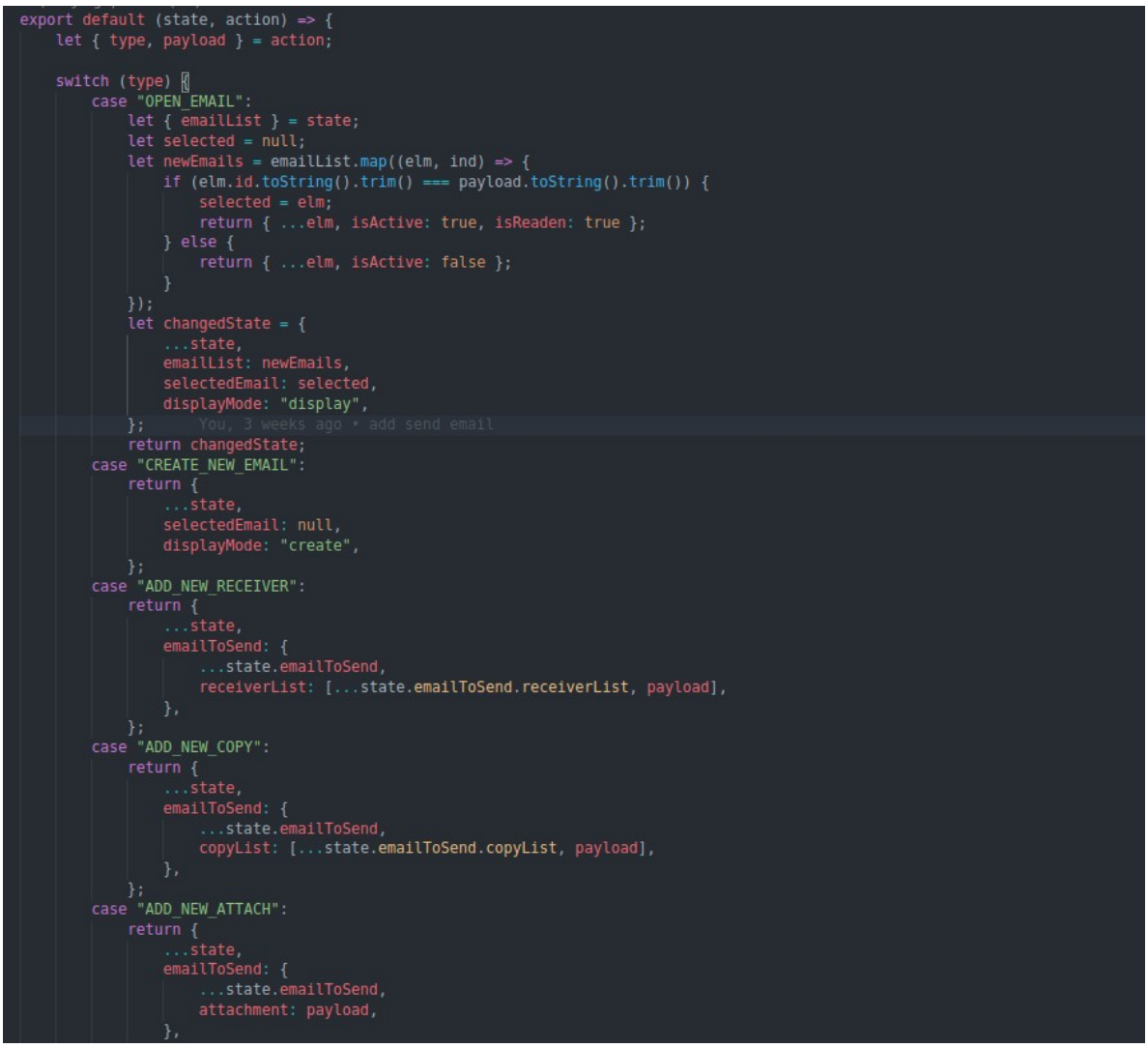
- displayMode répresente le mode d'affichage sa peut etre affichage normal ou creation d'un message
- emailToSend répresente le model d'un message à envoyé vers une backend

```
export const EmailProvider = ({ children }) => {
    const [state, dispatch] = useReducer(EmailReducer, initState);

    function openEmail(id) {
        dispatch({
            type: "OPEN_EMAIL",
            payload: id,
        });
        update(`/round/mails/set/read/${id}`).then(() => {});
    }
    function setEmailList(emails) {      You, 2 weeks ago • adding draft
        //console.log("Emails dispatch",emails)
        dispatch({
            type: "SET_EMAIL_LIST",
            payload: emails,
        });
    }

    function createNewEmail() {
        dispatch({
            type: "CREATE_NEW_EMAIL",
        });
    }
    function addNewReceiver(email) {
        dispatch({
            type: "ADD_NEW_RECEIVER",
            payload: email,
        });
    }
    function addNewCopy(email) {
        dispatch({
            type: "ADD_NEW_COPY",
            payload: email,
        });
    }
    function addNewAttachment(attach) {
        dispatch({
            type: "ADD_NEW_ATTACH",
            payload: attach,
        });
    }
    function deleteReceiver(id) {
        dispatch({
            type: "DELETE_RECEIVER",
            payload: id,
        });
    }
    function deleteCopy(id) {
        dispatch({
            type: "DELETE_COPY",
            payload: id,
        });
    }
```

- Cette figure répresente les deffirents fonctionalité qui permet de modifier l'etat globale de module selon le reducer

```jsx
return (
    <EmailManagementContext.Provider
        value={{
            emailList: state.emailList,
            selectedEmail: state.selectedEmail,
            openEmail,
            createNewEmail,
            addNewAttachment,
            addNewCopy,
            addNewReceiver,
            deleteAttachment,
            deleteReceiver,
            deleteCopy,
            addMailContent,
            addMailObj,
            addNewAttachment,
            setEmailList,
            resetMode,
            searchEmail,
            displayMode: state.displayMode,
            emailToSend: state.emailToSend,
            backupList: state.backupList,
        }}
    >
        <ContextDevTool
            context={EmailManagementContext}
            id="uniqContextId"
            displayName="EmailContext"
        />
        {children}
    </EmailManagementContext.Provider>
);
```

- tous les fonctions ainsi que les composants de l'etat initiale doit etre passer dans un objet ''Value'' du context .



```
export default (state, action) => {
    let { type, payload } = action;

    switch (type) {
        case "OPEN_EMAIL":
            let { emailList } = state;
            let selected = null;
            let newEmails = emailList.map((elm, ind) => {
                if (elm.id.toString().trim() === payload.toString().trim()) {
                    selected = elm;
                    return { ...elm, isActive: true, isReaden: true };
                } else {
                    return { ...elm, isActive: false };
                }
            });
            let changedState = {
                ...state,
                emailList: newEmails,
                selectedEmail: selected,
                displayMode: "display",
            };      You, 3 weeks ago • add send email
            return changedState;
        case "CREATE_NEW_EMAIL":
            return {
                ...state,
                selectedEmail: null,
                displayMode: "create",
            };
        case "ADD_NEW_RECEIVER":
            return {
                ...state,
                emailToSend: {
                    ...state.emailToSend,
                    receiverList: [...state.emailToSend.receiverList, payload],
                },
            };
        case "ADD_NEW_COPY":
            return {
                ...state,
                emailToSend: {
                    ...state.emailToSend,
                    copyList: [...state.emailToSend.copyList, payload],
                },
            };
        case "ADD_NEW_ATTACH":
            return {
                ...state,
                emailToSend: {
                    ...state.emailToSend,
                    attachment: payload,
                },
```

- cette figure réprésentent le ‘’Reducer’’ qui permet de gerer l’etat globale du contexte