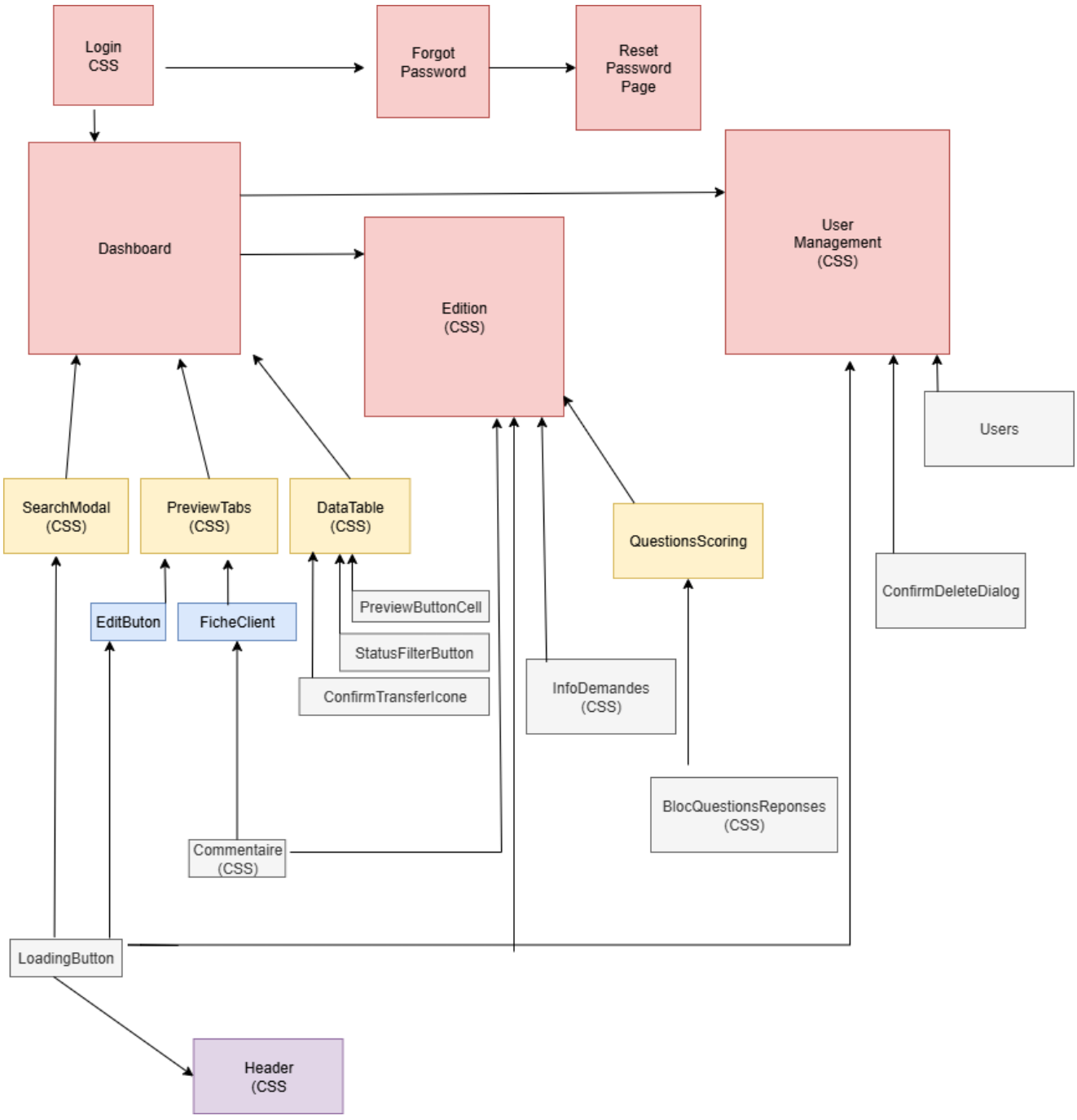
Documentation front-end application *qualilead*

Technologies utilisées : React + MUI + Bootstrap

Style : utilisation d'un fichier 'style.css' pour le style global, de fichiers css locaux (indiqués par '(CSS)' dans le diagramme), et css inline avec la prop sx dans les composants MUI.

Diagramme des pages et composants créés



A modifier pour les phases de validation et déploiement

STATUS

Mettre à jour les numéros id des statuts dans **la page Edition**, dans le **composant DataTable** et dans le **composant Dashboard**, et adapter les fonctions et les filtres (pour les affichages dans le tableau et dans les résultats de la recherche)

Page Edition

```
const STATUS_MAPPING = {

    4: 'En cours',
    6: 'Sans intérêt',
    7: 'A traiter'
  };
```

5= répondu

```
 const handleSaveAndSend = async () => {
  setSaveDialogOpen(false);
  handleSaveData(true, 5);
  await unlockQuote(selectedQuote.id); // shouldSend = true, status = 5
(envoyé)
};
```

4 = en cours

```
 const handleSaveAndReturnLater =async () => {
  setSaveDialogOpen(false);
  handleSaveData(false, 4);
  await unlockQuote(selectedQuote.id);// shouldSend = false, status = 4 (en
attente)
};
```

6 = sans intérêt

```
const handleConfirmSansInteret = async () => {
  setConfirmSansInteretOpen(false); // Ferme la popup
```

```
  if (!selectedQuote) {
    setMessage('Devis non trouvé');
    setMessageType('error');
    return;
  }

  try {
    const updatedData = { status: 6 };
    await editQuote(selectedQuote.id, updatedData);
    await unlockQuote(selectedQuote.id);

    setMessage('Demande marquée comme sans
```

Affichage conditionnel du bouton « sans intérêt »

```
 {Number(status) !== 6 && (<LoadingButton
              variant="contained"
              className="bouton bouton-fermer-edition"
              disabled={isLoading}
              onClick={() => setConfirmSansInteretOpen(true)} // Ouvre la
popup au lieu d'envoyer directement
              style={{ marginLeft: '10px' }}
            >
              sans intérêt
            </LoadingButton>)}
```

Composant DataTable

Filtrer pour éliminer les statuts répondus (= 5)

```
const getFilteredData = () => {
    let filtered = (data || []).filter(quote => quote.status !== 5);
```

Association du numéro id de statut à son libellé :

```
const STATUS_MAPPING = {
    4: 'En cours',
    6: 'Sans intérêt',
    7: 'A traiter'
  };
```

Fonction de transfert de la demande

```
 const handleTransfer = async (quoteId) => {
    try {
      await editQuote(quoteId, { status: 5 });
      setMessage('Transfert effectué avec succès')
```

Filtre pour les options de valeur de statut

```
 const getStatusValueOptions = () => {
    const allData = (data || []).filter(quote => quote.status !== 5);
    const uniqueStatusesInData = [...new Set(allData.map(item => item.st
```

Composant Dashboard

Filtre pour éliminer le satuts répondus (= 5)

```
//filtrage pour passer les données à la fonction de recherche
  const filteredDataSearch = useMemo(() => {
  return tableData?.filter(item => item.status !== 5) || [];
}, [tableData]);
```

```
<SearchModal
        open={searchModalOpen}
        onClose={closeSearchModal}
        onSearch={performSearch}
        tableData={filteredDataSearch}
      />
    </Box>
```

ROLES

Mettre à jour les numéros id des rôles / groupes dans les **composant UserManagement** et **Users**

Composant UserManagement

```
      setAvailableGroups([
        { id: 1, name: 'Gestionnaire Acceor' },
        { id: 2, name: 'Gestionnaire Options' },
        { id: 3, name: 'Utilisateur Acceor' },
        { id: 4, name: 'Utilisateur Options' },
      ]);
    } catch (err) {
      setError('Erreur chargement données');
    }
  };
  fetchData();
}, [token]);
```

Composant Users

```
const getAvailableRoles = () => {
  // Mapping basé sur les IDs réels de la bdd
  const groupMapping = {
    'Gestionnaire Acceor': '2',    // ID réel: 2
    'Gestionnaire Options': '1',   // ID réel: 1
    'Utilisateur Acceor': '4',     // ID réel: 4
    'Utilisateur Options': '3'     // ID réel: 3
  };
 const groupValue = groupMapping[currentUserGroup];

  switch(groupValue) {
    case '2': // Gestionnaire Acceor (ID=2)
      return [
        { value: '2', label: 'Gestionnaire Acceor' },
        { value: '4', label: 'Utilisateur Acceor' }
      ];
    case '1': // Gestionnaire Options (ID=1)
      return [
        { value: '1', label: 'Gestionnaire Options' },
        { value: '3', label: 'Utilisateur Options' }
      ];
    default:
      return [];
  }
};
```

PERMISSIONS

Pour que l'utilisateur courant puisse avoir accès à la section « Mes utilisateurs » dans la page « Gestion du compte », et pouvoir ainsi les créer / éditer / supprimer, il doit satisfaire une condition liée à ses permissions (et être gestionnaire). Dans le composant **UserManagement**, il faut adapter cette permission selon les paramètres de la base de donnée.

Composant UserManagement

La permission requise en l'état est 'add_user'. A modifier selon la permission requise.

```
const user = useCurrentUser();
const hasFullRights = user?.permissions && ['add_user'].every(p =>
user.permissions.includes(p));
```

COMMENTAIRES

Je n'ai accès qu'au champ « comment » des quotes. Je n'ai donc pas pu finaliser la différenciation des deux champs commentaire (commentaire sur la demande et commentaire sur l'échange avec le client).

Composant FicheClient

Mettre les bonnes valeurs (la valeur pour les deux commentaires est actuellement 'selectedQuote ?.comment')

```
1. <Box className="container-commentaires">
2.
3.             <Commentaire className="champ" value={selectedQuote?.comment
   || " "}
4.                     onChange={() => {}} editable = {false}
   label={"Commentaire précisant la demande"}/>
5.
6.         {/*valeur de la prop value à changer selon la clé du champ */}
7.             <Commentaire className="champ" value={selectedQuote?.comment
   || " "}
8.                     onChange={() => {}} editable = {false}
   label={"Commentaire précisant l'échange"}/>
9.         </Box>
```

Composant Edition

Modifier la fonction fomatDataInfoForApi en ajoutant le paramètres commentaireEchange, et dans le return indiquer ce qui convient (actuellement il y a seulement 'comment : commentaireDemande')

```
// Fonction pour formater les données info demande selon l'API

function formatDataInfoForApi(blocInfoData, commentaireDemande) {
  return {
    order_id: blocInfoData.order_id,
    reference: blocInfoData.reference,
    firstname: blocInfoData.Prénom,
    lastname: blocInfoData.Nom,
    phone: blocInfoData["Numéro de téléphone"],
    customer_email: blocInfoData.Email,
    weeknumber: parseInt(blocInfoData["Semaine N°"]),
    call_count: parseInt(blocInfoData["Nombre d'appels"]),
    date_first_call: blocInfoData["Date du 1er appel"],
    date_last_call: blocInfoData["Date du dernier appel"],
    idEtablissement: blocInfoData.idEtablissement,
```

```
    reference: parseInt(blocInfoData.Référence),
    status: parseInt(blocInfoData.status),
    comment: commentaireDemande || ""
  };

}
```

Pour les deux types de commentaire, distinguer deux états et deux fonctions: enlever le commentaire pour le second type et modifier les valeurs en fonction des noms des clés (actuellement 'selectedQuote.comment' pour les deux types) :

```
const [commentaireDemande, setCommentaireDemande] = useState("");
 useEffect(() => {
 if (selectedQuote && selectedQuote.comment) {
   setCommentaireDemande(selectedQuote.comment);
 }
}, [selectedQuote]);
/*const [commentaireEchange, setCommentaireEchange] = useState("");
 useEffect(() => {
  if (selectedQuote && selectedQuote.comment) {
    setCommentaireEchange(selectedQuote.comment);
  }
}, [selectedQuote]); */
```

```
 const handleCommentaireDemandeChange = (newValue) => {
    setCommentaireDemande(newValue);
  };

  /*const handleCommentaireEchangeChange = (newValue) => {
    setCommentaireEchange(newValue);
  }; */
```

Dans la fonction handleSaveData :

Ajouter le commentaireEchange dans l'utilisation de la fonction formatDataInfoForApi

```
 try {
    const dataInfoToSend = formatDataInfoForApi(blocInfoData,
commentaireDemande);
```

Dans les composants « Commentaire » :

Remplacer dans le second « commantaireDemande » par « commentaireEchange » et
« handlecomentaireDemangeChange » par « handleCommentaireEchangeChange »

```
{/* Ligne Commentaires */}
        <Row className="mb-4">
          <Col xs={6} className="d-flex">
            <Commentaire
              value={commentaireDemande}
              onChange={handleCommentaireDemandeChange}
              label={"Commentaire précisant la demande du client"}
            />
            </Col>
          <Col xs={6} className="d-flex">

            {/*props à modifier lorsqu'on aura le libellé du champ commentaire
demande */}
            <Commentaire
              value={commentaireDemande}
              onChange={handleCommentaireDemandeChange}
              label={"Commentaire précisant l'échange avec le client"}
            />
          </Col>
```

DIVERS

- Supprimer la page changePassword et la route qui correspond ? Il y a déjà les pages forgot-password et reset-password
- Supprimer la page cards ? Je ne suis pas sûr qu'elle soit utile.
- Vérifier que date du premier appel soit antérieure à date du second appel ?
- Verifier que réponses date de votre evenement est elle fixée est cohérente avec quelle est la date de l'événement ?
- **En l'état de mon environnement, si l'utilisateur est inactif mais qu'il fait une demande de réinitialisation de mot de passe, cela fonctionne, même si après il ne peut pas se connecter (identifiant invalide)**
- **En l'état de mon environnement, les permissions coté back-end sont mal paramétrée. Si on n'est pas coché super utilisateur côté back end, cela crée des erreurs de permission pour créer / supprimer/ editer un utilisateur selon qu'il est ou non gestionnaire ou utilisateur (erreur 403)**
- Créer la possibilité de décocher une case précohée
- Lorsque des erreurs dans la création de l'utilisateur (ex erreur 400) préciser la raison ? Ex l'utilisateur existe déjà dans la base de donnée mais a été désactivé
- Lorsqu'on veut changer son mot de passe (dans la page Gestion du compte) préciser les erreur et pas simpelment retour Erreur http : 400 ? EN particulier si le format du mot de passe n'est pas valide (ce qui peut créer l'erreur 400 ?)
- Le responsive : le header et certaines fenêtres / bouton n'ont pas toujours un rendu correct pour les versions mobiles.