# Table des matières

[Table des matières 1](#_Toc59266887)

[1 Cahier des charges 4](#_Toc59266888)

[1.1 Cadre 4](#_Toc59266889)

[1.2 Application mobile 4](#_Toc59266890)

[1.3 Site Web 5](#_Toc59266891)

[2 Schémas généraux 6](#_Toc59266892)

[2.1 Application mobile 6](#_Toc59266893)

[2.2 Application web 7](#_Toc59266894)

[3 Analyse fonctionnelle 8](#_Toc59266895)

[3.1 Application Mobile 8](#_Toc59266896)

[3.1.1 Accueil 8](#_Toc59266897)

[3.1.2 Menu 9](#_Toc59266898)

[3.1.3 Recherche 10](#_Toc59266899)

[3.1.4 Liste des Résultats 11](#_Toc59266900)

[3.1.5 Recherche Photo 12](#_Toc59266901)

[3.1.6 Non trouvée 13](#_Toc59266902)

[3.1.7 Bière 14](#_Toc59266903)

[3.1.8 Nouvelle bière 16](#_Toc59266904)

[3.1.9 Favoris 18](#_Toc59266905)

[3.1.10 Où trouver 20](#_Toc59266906)

[3.1.11 Suggestions 23](#_Toc59266907)

[3.1.12 Etablissement/brasserie 25](#_Toc59266908)

[3.1.13 Liste Bière Etablissement 26](#_Toc59266909)

[3.2 Site Web 27](#_Toc59266910)

[3.2.1 Accueil 27](#_Toc59266911)

[3.2.2 Register 28](#_Toc59266912)

[3.2.3 Tableau de bord Achats 29](#_Toc59266913)

[3.2.4 Achat 30](#_Toc59266914)

[3.2.5 Confirmation achats 31](#_Toc59266915)

[3.2.6 Factures 32](#_Toc59266916)

[3.2.7 Tableau De Bord Plans tarifaires 33](#_Toc59266917)

[3.2.8 Tableau de bord Etablissement 34](#_Toc59266918)

[3.2.9 Ajout/modification de bières a un établissement 35](#_Toc59266919)

[3.2.10 Création/Modification bière 36](#_Toc59266920)

[3.2.11 Création/Modification Etablissement 37](#_Toc59266921)

[3.2.12 Tableau de bord utilisateur 38](#_Toc59266922)

[3.2.13 Gestion Utilisateurs 39](#_Toc59266923)

[3.2.14 Gestion des rôles 40](#_Toc59266924)

[3.2.15 Modification Rôles d’un GroupUser 41](#_Toc59266925)

[3.2.16 Modification des utilisateurs ayant un rôle 42](#_Toc59266926)

[3.2.17 Modification Accès GroupAchat 43](#_Toc59266927)

[3.2.18 Modification Accès GroupBière 44](#_Toc59266928)

[3.2.19 Modification Accès GroupEtablissement 45](#_Toc59266929)

[3.2.20 Tableau de bord des bières 46](#_Toc59266930)

[3.3 IdentityServer 47](#_Toc59266931)

[3.3.1 Application Mobile 47](#_Toc59266932)

[4 Contraintes fonctionnelles 50](#_Toc59266933)

[4.1 Règles d’accès et autorisations 50](#_Toc59266934)

[4.1.1 Application mobile 50](#_Toc59266935)

[4.1.2 Site Web 50](#_Toc59266936)

[4.2 Règles de structure 52](#_Toc59266937)

[4.2.1 Bière 52](#_Toc59266938)

[4.2.2 TypeBière 52](#_Toc59266939)

[4.2.3 AvisBière 52](#_Toc59266940)

[4.2.4 Organisation 53](#_Toc59266941)

[4.2.5 TypeOrganisation 53](#_Toc59266942)

[4.2.6 Facture 53](#_Toc59266943)

[4.2.7 Adressefacturation 53](#_Toc59266944)

[4.2.8 Etablissement 53](#_Toc59266945)

[4.2.9 Typeétablissement 53](#_Toc59266946)

[4.2.10 Horaire 54](#_Toc59266947)

[4.2.11 Utilisateur 54](#_Toc59266948)

[4.3 Règles de validation 54](#_Toc59266949)

[4.3.1 Utilisateur 54](#_Toc59266950)

[4.3.2 Client 54](#_Toc59266951)

[4.3.3 GroupAdmin 54](#_Toc59266952)

[4.3.4 Organisation 54](#_Toc59266953)

[4.3.5 Bière 55](#_Toc59266954)

[4.3.6 Brasserie 55](#_Toc59266955)

[4.3.7 Horaire 55](#_Toc59266956)

[4.3.8 Avis 55](#_Toc59266957)

[4.4 Règles de calcul 55](#_Toc59266958)

[4.4.1 Vues 55](#_Toc59266959)

[4.4.2 Classement des bières 55](#_Toc59266960)

[4.4.3 Suggestions 56](#_Toc59266961)

[5 Schéma Entité-Association 57](#_Toc59266962)

[6 Schéma relationnel 58](#_Toc59266963)

[7 Entités 59](#_Toc59266964)

[7.1 Bières 59](#_Toc59266965)

[8 Analyse technique 60](#_Toc59266966)

[8.1 Technologies proposées 60](#_Toc59266967)

[8.1.1 Bases de données 60](#_Toc59266968)

[8.1.2 API 60](#_Toc59266969)

[8.1.3 Site Web 60](#_Toc59266970)

[8.1.4 Application Mobile 62](#_Toc59266971)

[8.1.5 Authentification 62](#_Toc59266972)

[8.2 Architecture applicative 62](#_Toc59266973)

# Cahier des charges

## Cadre

Le client souhaite développer une application mobile « BeerUp » et un site web.

L’application mobile sera à destination du public. À travers cette application, le public pourra noter les bières selon leurs saveurs et donner une note générale ainsi qu’obtenir des suggestions et trouver où acheter/consommer une bière. Le site web sera à destination de ses clients qui pourront à travers celui-ci acheter des vues sponsorisées pour faire ressortir leur bière et/ou établissement.

Une vue sponsorisée correspond à l’affichage d’une bière ou d’un établissement en haut de la liste de recherche ou de suggestion par l’utilisateur.

## Application mobile

Le client souhaite développer une application mobile permettant à l’utilisateur de chercher une bière via texte ou photographie d’une bouteille, de la commenter, de la noter et de la réserver dans une bibliothèque « bières favorites »

L’utilisateur doit pouvoir rechercher une bière par son nom ou encore la liste des bières brassées par un brasseur spécifique.

L’utilisateur doit pouvoir à travers une photo réalisée via l’application pour chercher une bière

Si la bière n’existe pas, l’utilisateur peut ajouter la bière à la base de données

Si l’utilisateur a déjà noté la bière, la page affiche les notes pour modifications éventuelles.

Une fois la bière trouvée l’utilisateur peut donner une note générale de 0 à 5 ainsi que spécifier et noter les différents goûts selon une échelle de 0 à 5. Les différents goûts sont :

1. Amertume
2. Houblonné
3. Caramel
4. Café
5. Malté
6. Sucré
7. Acide
8. Fruité

La moyenne de la note générale donnée à la bière par les autres utilisateurs est également affichée

Une option propose de trouver une bière similaire

Une option est également proposée d’ajouter la bière à la bibliothèque des favoris de l’utilisateur.

Dans sa bibliothèque, l’utilisateur doit pouvoir supprimer ses favoris ainsi que retourner vers les bières présentes.

L’utilisateur peut demander des suggestions basées :

* Selon un algorithme qui reprend les bières de même type déjà dans la bibliothèque de l’utilisateur et les meilleurs classements du mois
* Selon un algorithme aléatoire des bières avec les meilleures notes du mois
* Selon un algorithme « surprise »
* Dans tous les cas, la première proposition sera une proposition « annonceur » choisie aléatoirement parmi les bières pour lesquelles des vues ont été achetées par un client.

Affichage pour la bière trouvée d’une moyenne de 0 à 5 ainsi que la moyenne des gouts

L’application permet également de faire une recherche sur la page d’une bière avec une carte et localisation de l’utilisateur et d’afficher une page info : (dans un rayon de 20 km)

* Les magasins qui vendent la bière trouvée
* Les bars qui servent la bière trouvée
* Le premier résultat des 2 recherches est un résultat annonceur avec une vue sponsorisée s’il y en a une dans les 20km et choisi aléatoirement s’il y en a plusieurs.

Une page avec une photo, l’adresse, l’horaire et l’adresse web du bar/magasin est affichée lorsque l’on clique sur le marqueur sur la carte.

Le marqueur sur la carte aura une couleur différente en fonction du fait que le bar/magasin est ouvert ou fermé.

L’utilisateur peut également voir la page d’info du brasseur si elle existe

## Site Web

Les clients sont regroupés par groupe, chaque groupe peut avoir des établissements, des bières liées aux établissements et éventuellement des bières liées à leur brasserie.

Seules les bières liées à une brasserie appartenant au groupe peuvent faire l’objet d’achat de vue.

Le site se divise en 3 parties : accueil, administration et contact.

La partie administration se découpe en :

* Gestion des utilisateurs
* Gestion des bières
* Gestion des établissements
* Gestion des achats

Dans la partie gestion des utilisateurs, il doit être possible de changer le rôle des utilisateurs et d’affiner les accès que ceux-ci ont aux différentes parties.

Dans la partie gestion des bières, il doit être possible de modifier les bières liées au groupe et d’en créer de nouvelles.

Dans la partie gestion des établissements et brasseries il doit être possible de modifier les établissements et brasseries liés au groupe du client et d’en créer de nouveaux.

Dans la partie gestion des achats, il doit être possible de visualiser le nombre d’achats de vues sponsorisées réalisés par produit et établissement ainsi que les vues sponsorisées visualisées. De plus, le client doit pouvoir acheter de nouvelles vues sponsorisées et exporter ses factures.

L’administrateur doit pouvoir gérer tous les utilisateurs, bières et établissements. De plus une vue des ventes mensuelles avec des sous-totaux par groupe doit exister.

# Schémas généraux

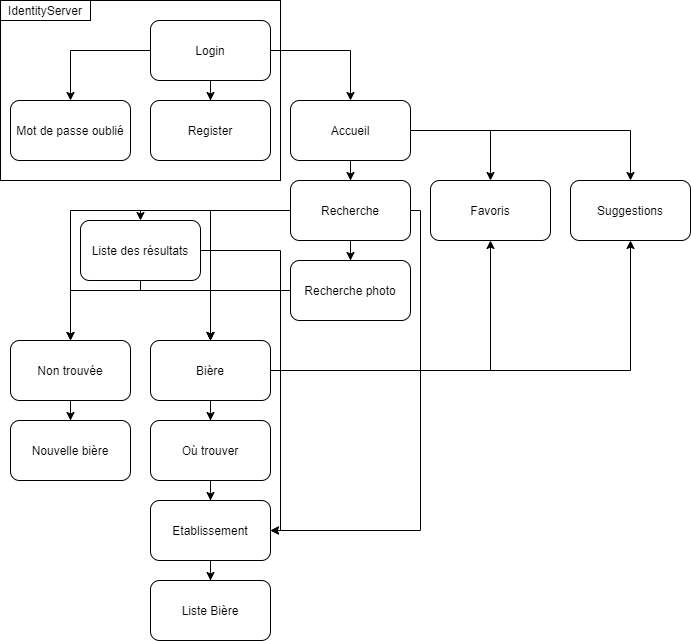
Un schéma très global de l’application mobile ainsi que l’application web est présenté lors de ce chapitre ainsi que la logique sous-jacente au diagramme.

## Application mobile

L’idée de l’application est que l’utilisateur enregistré puisse facilement trouver une bière ou un établissement.

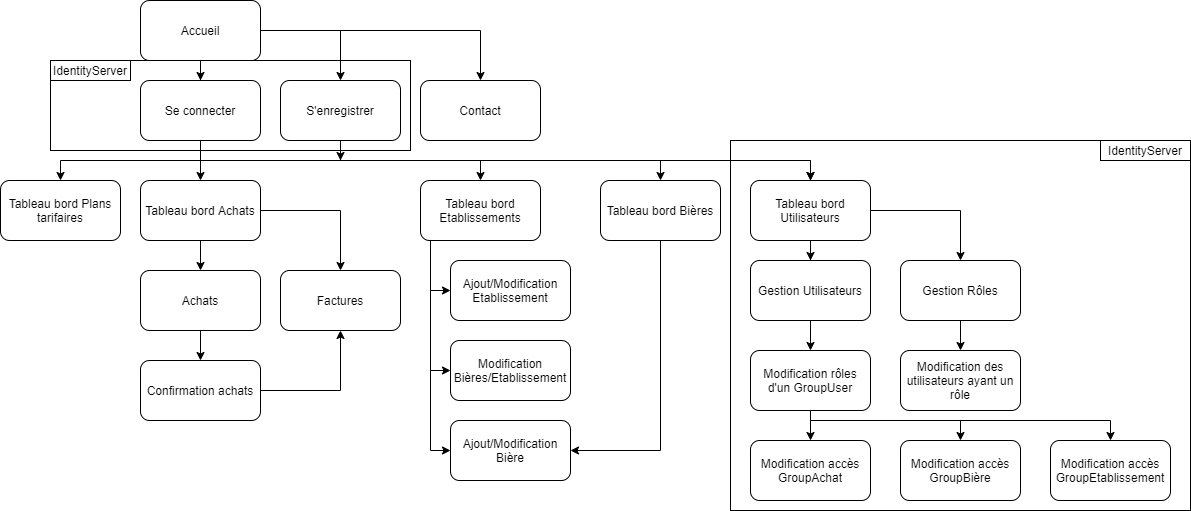
Il faut également que l’utilisateur trouve facilement les suggestions pour visualiser les vues sponsorisées.

L’objectif étant de récolter un maximum d’avis pour créer l’envie d’aller chercher des suggestions ou des lieux dans lesquels la bière est servie/vendue.



## Application web

L’idée est qu’un GroupOwner puisse facilement gérer les différents aspects nécessaires et puisse être très flexible dans sa gestion des utilisateurs et de leurs rôles.



# Analyse fonctionnelle

## Application Mobile

### Accueil

#### Ecran



#### Description textuelle

Résumé : écran d’accueil de l’application mobile, il affiche la barre de raccourcis vers les différents menus ainsi qu’une bière sélectionnée parmi celles avec des vues sponsorisées disponibles.

Acteurs : Utilisateurs *(primaire)* l’Application Mobile *(secondaire)* et l’API Bières *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile

Scénario nominal :

1. L’application mobile demande une bière qui au moins une vue sponsorisée disponible à l’API Bières
2. L’API Bières sélectionne une bière
3. L’API Bières ajoute une vue à la bière dans la base de données
4. L’API Bières renvoie la bière à l’Application Mobile
5. L’application mobile affiche les détails de la bière sur la page d’accueil
6. L’utilisateur clique sur un raccourci ou sur la bière du jour.
7. L’utilisateur est envoyé vers l’écran correspondant

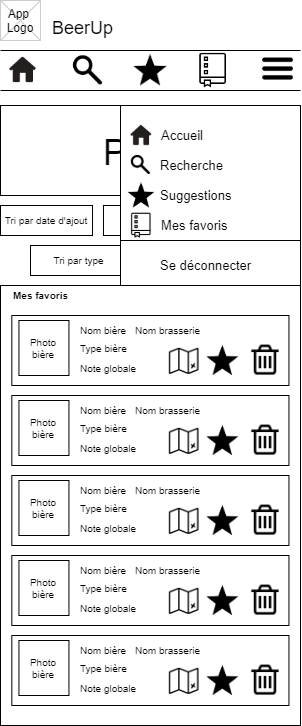
Scénarios alternatifs :

*A1 :* Aucune bière n’a de vue sponsorisée disponible

1. L’API Bières sélectionne aléatoirement une bière du top 10 du mois (30 jours calendrier) des notes générales
2. L’API Bières renvoie la bière à l’Application Mobile

### Menu

#### Ecran



#### Description textuelle

Résumé : Le menu en haut à droite permet :

* De retourner à l’écran d’accueil
* D’aller à l’écran de recherche
* D’aller à l’écran des suggestions
* D’aller à l’écran « mes favoris »
* De se déconnecter

### Recherche

#### Ecran



#### Description textuelle

Résumé : écran de recherche de l’Application Mobile, il permet à l’utilisateur de faire soit une recherche textuelle soit de faire le lien vers la recherche par photo.

Acteurs : Utilisateurs *(primaire)* l’Application Mobile *(secondaire),* l’API Bières *(secondaire),* l’API Établissements *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile et est sur la page recherche

Scénario nominal :

1. L’utilisateur entre le nom d’une bière ou d’une brasserie
2. L’utilisateur clique sur le bouton recherche (logo loupe)
3. L’application mobile effectue une demande aux API Bières et API Établissements
4. Les 2 API répondent les résultats qui correspond au moins au texte encodé
5. L’application mobile redirige l’utilisateur vers le résultat de la recherche (fiche de la bière ou de l’établissement en question)

Scénarios alternatifs :

*A1 :* L’utilisateur clique sur la recherche par photo

1. L’application mobile redirige l’utilisateur vers l’écran de recherche par photo

*A2 :* Plusieurs résultats correspondent à la recherche

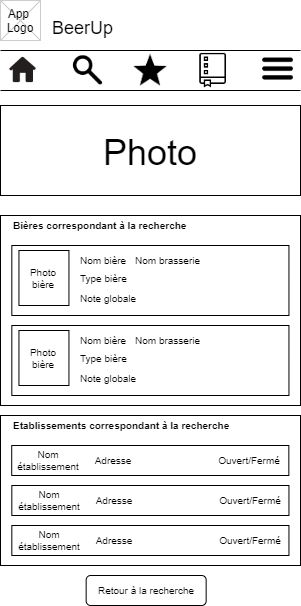
1. L’application mobile redirige l’utilisateur vers l’écran de résultats

*A3 :* Aucun résultat ne correspond à la recherche

1. L’application mobile redirige l’utilisateur vers l’écran de non-trouvé

### Liste des Résultats

#### Ecran



#### Description textuelle

Résumé : écran de la liste des résultats après une recherche par l’utilisateur, il permet à l’utilisateur de cliquer sur une bière ou sur un établissement pour visualiser la fiche en question.

Acteurs : Utilisateurs *(primaire)* l’Application Mobile *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile, a effectué une recherche et plusieurs résultats sont sortis de sa recherche

Scénario nominal :

1. L’Application Mobile reçoit les objets de la recherche
2. L’Application Mobile affiche les différents résultats
3. L’utilisateur clique sur le résultat qui l’intéresse
4. L’application mobile redirige l’utilisateur vers le résultat de la recherche (fiche de la bière ou de l’établissement en question)

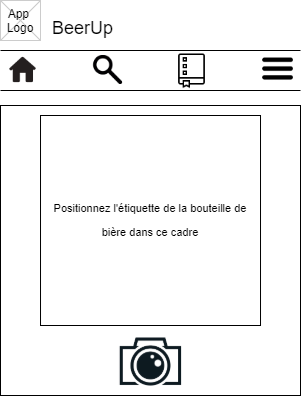
Scénarios alternatifs :

*A1 :* L’utilisateur clique sur le bouton « retour à la recherche »

1. L’application mobile redirige l’utilisateur vers l’écran de recherche

### Recherche Photo

#### Ecran



#### Description textuelle

Résumé : écran de prise de photo permettant à l’utilisateur de cadrer sa photo et l’envoyer pour reconnaissance.

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Vision *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile, a effectué une recherche et souhaité faire une recherche par photo
* L’utilisateur a donné les droits d’autorisation de prise de photo par l’Application Mobile

Scénario nominal :

1. L’utilisateur réalise la photo
2. L’Application Mobile transmet l’image à l’API Vision
3. L’application mobile redirige l’utilisateur vers le résultat de la recherche (fiche de la bière en question)

Scénarios alternatifs :

*A1 :* Plusieurs résultats correspondent à la recherche

1. L’application mobile redirige l’utilisateur vers l’écran de résultats

*A2 :* Aucun résultat ne correspond à la recherche

1. L’application mobile redirige l’utilisateur vers l’écran de non-trouvé

### Non trouvée

#### Ecran



#### Description textuelle

Résumé : écran informant l’utilisateur qu’une recherche n’a pas trouvé de résultat et proposant à celui-ci de rajouter une bière ou de refaire une nouvelle recherche.

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile, a effectué une recherche et celle n’a pas donné de résultats

Scénario nominal :

1. L’utilisateur clique sur le bouton « nouvelle recherche »
2. L’application mobile redirige l’utilisateur vers l’écran de recherche

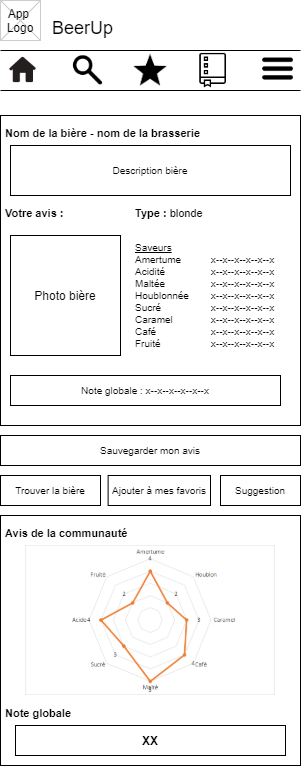
Scénarios alternatifs :

*A1 :* L’utilisateur souhaite rajouter une nouvelle bière

1. L’application mobile redirige l’utilisateur vers l’écran de création d’une bière

### Bière

#### Ecran



#### Description textuelle

Résumé : écran détaillant toutes les informations à propos d’une bière.

* La première partie de l’écran affiche les notes que l’utilisateurs aurait déjà donné à la bière.
* La deuxième affiche la moyenne de ces valeurs données par l’ensemble des utilisateurs.
  + Les notes des saveurs sont affichées en graphique type « radar »
  + La note globale est affichée avec 2 décimales

L’utilisateur peut également modifier son avis dans cet écran, trouver les endroits où se vendent la bière, rajouter la bière à ses favoris ou demander une suggestion basée sur la bière

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Bières *(secondaire),*

Préconditions :

* L’utilisateur a lancé l’application mobile, a souhaité voir le détail d’une bière.

Scénario nominal :

1. L’Application Mobile fait appel à l’Api Bière pour obtenir l’avis de l’utilisateur sur la bière
2. L’Api Bières renvoie l’avis de l’utilisateur
3. L’Application Mobile fait appel à l’Api Bière pour obtenir la moyenne des différents avis sur la bière
4. L’Api Bières renvoie les moyennes
5. L’Application Mobile affiche les résultats
6. L’utilisateur modifie ses notes
7. L’utilisateur appuie sur le bouton « sauvegarder mon avis »
8. L’Application Mobile envoie à l’Api Bière l’avis de l’utilisateurs sur la bière pour une modification
9. Une notification est affichée à l’utilisateur pour l’informer que l’avis est bien sauvegardé

Scénarios alternatifs :

*A1 :* L’utilisateur n’a pas encore donné sur la bière

1. L’avis affiché de l’utilisateur est initialisé à 0

*A2 :* L’utilisateur souhaite rajouter la bière à ses favoris

1. L’Application Mobile envoie à l’Api Bière l’identifiant de la bière et de l’utilisateur à l’API Bière pour ajouter la bière aux favoris de l’utilisateur
2. Une notification est affichée à l’utilisateur pour l’informer que la bière est bien rajoutée à ses favoris

*A3 :* L’utilisateur souhaite savoir où trouver la bière

1. L’application mobile redirige l’utilisateur vers l’écran « où trouver »

*A4 :* L’utilisateur souhaite avoir une suggestion basée sur cette bière

1. L’application mobile redirige l’utilisateur vers l’écran « suggestions »

Enchainements d’erreurs :

*E1 :* L’utilisateur encode moins de 2 notes de saveurs avec une valeur différente à 0

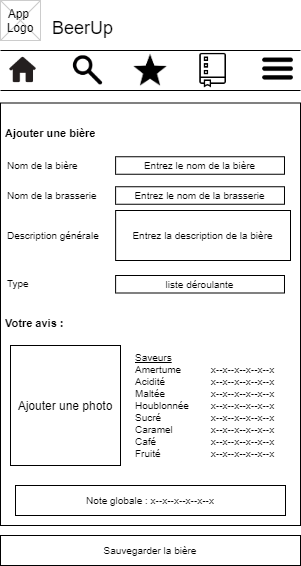
1. Un message s’affiche pour prévenir l’utilisateur qu’il faut au moins 2 valeurs différentes de 0

*E2 :* L’utilisateur a déjà donné 2 fois un avis sur la bière aujourd’hui

1. Un message s’affiche pour prévenir l’utilisateur qu’il ne peut donner son avis que 2 fois

### Nouvelle bière

#### Ecran



#### Description textuelle

Résumé : écran permettant à l’utilisateur de rajouter une bière à la base de données.

Les champs « nom de la biére », « nom de la brasserie », « type » ainsi que l’avis et la note globale sont obligatoires. La description et la photo sont des champs optionnels

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Bières *(secondaire),*

Préconditions :

* L’utilisateur a lancé l’application mobile et a souhaité ajouter une bière

Scénario nominal :

1. L’utilisateur encode les différentes informations
2. L’utilisateur clique sur le bouton « sauvegarder la bière »
3. L’Application Mobile envoie à l’Api Bière les données encodées
4. L’application mobile redirige l’utilisateur vers la fiche de la bière nouvellement créée

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite rajouter une photo

1. L’utilisateur est invité à rajouter une photo de la bière via l’ouverture de la caméra de son téléphone

Enchainements d’erreurs :

*E1 :* L’utilisateur encode moins de 2 notes de saveurs avec une valeur différente à 0

1. Un message s’affiche pour prévenir l’utilisateur qu’il faut au moins 2 valeurs différentes de 0

*E2 :* L’utilisateur a déjà donné 2 fois un avis sur la bière aujourd’hui

1. Un message s’affiche pour prévenir l’utilisateur qu’il ne peut donner son avis que 2 fois

*E3 :* L’utilisateur a déjà créé 5 bières aujourd’hui

1. Un message s’affiche pour prévenir l’utilisateur qu’il ne peut créer que 5 nouvelles bières par jour

*E4 :* L’utilisateur n’a pas rempli tous les champs obligatoires

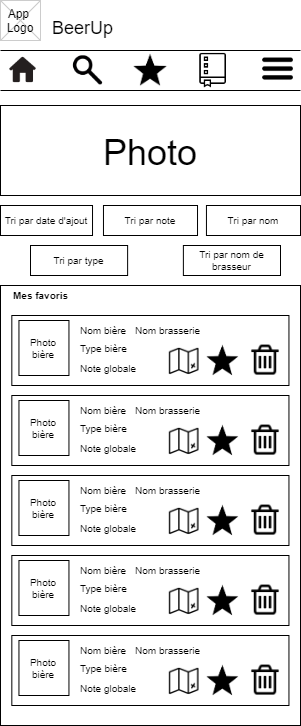
1. Un message de validation s’affiche à côté du champ à remplir pour demander le remplissage

*E5 :* L’utilisateur a rempli le champ description avec plus de 1000 caractères

1. Un message de validation s’affiche à côté du champ pour prévenir l’utilisateur de la longueur maximale du champ

### Favoris

#### Ecran



#### Description textuelle

Résumé : écran permettant à l’utilisateur de consulter ses bières favorites, de trouver où les acheter, de demander une suggestion ou encore de les supprimer de la liste des favoris. Par défaut les bières sont triées par ordre d’ajout le plus récent au moins récent.

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Bières *(secondaire),*

Préconditions :

* L’utilisateur a lancé l’application mobile et a souhaité voir ses favoris

Scénario nominal :

1. L’Application Mobile fait appel à l’Api Bière pour obtenir les bières favorites de l’utilisateur
2. L’Api Bières renvoie la liste des bières
3. L’Application Mobile affiche les résultats

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite savoir où trouver la bière

1. L’application mobile redirige l’utilisateur vers l’écran « où trouver »

*A2 :* L’utilisateur souhaite avoir une suggestion basée sur cette bière

1. L’application mobile redirige l’utilisateur vers l’écran « suggestions »

*A3 :* L’utilisateur souhaite voir la fiche d’une bière

1. L’application mobile redirige l’utilisateur vers l’écran « bière »

*A4 :* L’utilisateur souhaite supprimer une bière de ses favoris

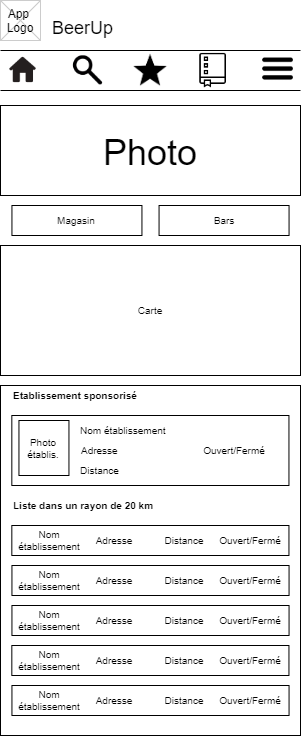
1. L’Application Mobile fait appel à l’Api Bière pour supprimer la bière des favoris de l’utilisateur
2. L’Api Bières confirme la suppression
3. L’Application Mobile affiche une notification à l’utilisateur

*A5 :* L’utilisateur souhaite changer le type de tri des favoris

1. L’application mobile modifie l’ordre du tri selon le paramètre choisi par l’utilisateur

### Où trouver

#### Ecran



#### Description textuelle

Résumé : écran permettant à l’utilisateur de visualiser les différents établissements vendant la bière dans un rayon de 20 km aussi bien sous forme de marqueur sur une carte que sous la forme d’une liste.

Chaque ligne reprend le nom, l’adresse, la distance de l’établissement et s’il est ouvert ou fermé. Les établissements non payants sont triés par statut d’ouverture et puis par distance.

Ces établissements sont repris sur la carte avec des marqueurs de couleurs selon leur statut, rouge pour « fermé » et vert pour « ouvert ».

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Établissements *(secondaire),*

Préconditions :

* L’utilisateur a lancé l’application mobile et a souhaité trouver les points de vente d’une bière

Scénario nominal :

1. L’Application Mobile récupère la position GPS de l’utilisateur
2. L’Application Mobile fait appel à l’Api Établissements pour obtenir la liste des établissements
3. L’Api Établissements renvoie la liste des établissements ainsi que l’établissement avec vue sponsorisée et ouvert
4. L’API Établissements ajoute une vue à l’établissement qui a la vue sponsorisée dans la base de données
5. L’Application Mobile affiche en première ligne l’établissement avec une vue sponsorisée
6. L’Application Mobile affiche ensuite les autres établissements
7. L’Application Mobile affiche les établissements sur la carte avec des marqueurs
8. L’utilisateur clique sur un établissement
9. L’application Mobile redirige l’utilisateur vers l’écran « établissement »

Scénarios alternatifs :

*A1 :* L’utilisateur clique sur un marqueur d’établissement sur la carte

1. L’application affiche le nom de l’établissement et un lien vers sa fiche
2. L’utilisateur clique sur le lien de l’établissement
3. L’application Mobile redirige l’utilisateur vers l’écran « établissement »

*A2 :* Il existe plusieurs établissements avec vues sponsorisées ouverts (ou fermé le cas échéant – *voir E2 ci-dessous*)

1. L’Api Établissements sélectionne aléatoirement un établissement de la liste de ceux qui remplissent les critères
2. L’Api Établissements renvoie l’établissement

*A3 :* L’utilisateur souhaite afficher uniquement les bars

1. L’utilisateur clique sur le bouton « bar »
2. Le bouton « bar » devient le bouton « tous les résultats »
3. L’Application Mobile affiche uniquement les bars dans la liste et sur la carte

*A4 :* L’utilisateur souhaite afficher uniquement les magasins

1. L’utilisateur clique sur le bouton « magasins »
2. Le bouton « magasins » devient le bouton « tous les résultats »
3. L’Application Mobile affiche uniquement les magasins dans la liste et sur la carte

*A5 :* L’utilisateur souhaite réafficher toute la liste après une sélection

1. L’utilisateur clique sur le bouton « tous les résultats »
2. Les boutons sont réinitialisés à leur état initial
3. L’Application Mobile affiche tous les résultats dans la liste et sur la carte

Enchainements d’erreurs :

*E1 :* Il n’existe pas d’établissement avec vues sponsorisées disponibles dans un rayon de 20km

1. Aucun établissement sponsorisé n’est affiché

*E2 :* Il n’existe pas d’établissement avec vues sponsorisées ouvert

1. L’Api Établissements renvoie l’établissement avec vues sponsorisées et fermé

*E3 :* Après un filtre « bars », la vue sponsorisée n’est plus affichée car c’est un « magasin »

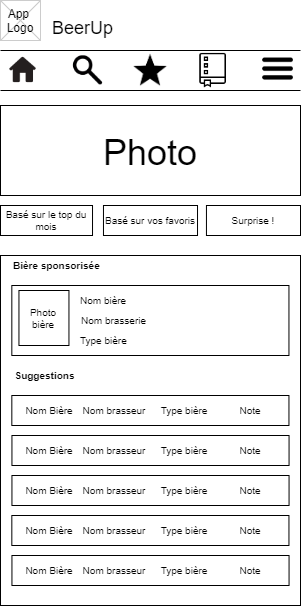
1. Aucune vue sponsorisée n’est affichée

*E4 :* Après un filtre « magasin », la vue sponsorisée n’est plus affichée car c’est un « bar »

1. Aucune vue sponsorisée n’est affichée

### Suggestions

#### Ecran



#### Description textuelle

Résumé : écran permettant à l’utilisateur de consulter des suggestions de bières. La liste des suggestions sont triées par ordre décroissant des notes globales.

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Bières *(secondaire),*

Préconditions :

* L’utilisateur a lancé l’application mobile et a souhaité une suggestion de bière sans passer par une bière au préalable ou l’utilisateur souhaite des suggestions basées sur le top du mois.

Scénario nominal :

1. L’Application Mobile fait appel à l’Api Bières pour obtenir une bière sponsorisée
2. L’Api Bières renvoie une bière sélectionnée aléatoirement parmi les bières avec des vues disponibles
3. L’API Bières ajoute une vue à la bière qui a la vue sponsorisée dans la base de données
4. L’Application Mobile fait appel à l’Api Bières pour obtenir une liste de 5 bières suggérées avec le critère :
   1. Bière qui est dans le top 10 des notes des utilisateurs du mois
5. L’Api Bières renvoie la liste de bières
6. L’Application Mobile affiche les résultats

Scénarios alternatifs :

*A1 :* L’utilisateur a souhaité voir des suggestions basées sur une bière préalablement sélectionnée

1. L’Application Mobile fait appel à l’Api Bières pour obtenir une bière sponsorisée
2. L’Api Bières renvoie une bière sélectionnée aléatoirement parmi les bières avec des vues disponibles et du même type que la bière préalablement sélectionnée
3. L’API Bières ajoute une vue à la bière qui a la vue sponsorisée dans la base de données
4. L’Application Mobile fait appel à l’Api Bières pour obtenir une liste de 5 bières suggérées avec les deux critères :
   1. Bière du même type
   2. Bière n’ayant pas de note trop éloignée pour chaque goût (+1/–1)
5. L’Api Bières renvoie la liste de bières
6. L’Application Mobile affiche les résultats

*A2 :* l’utilisateur souhaite des suggestions basées sur ses bières favorites

1. L’Application Mobile fait appel à l’Api Bières pour obtenir une liste de 5 bières suggérées avec les critères :
   1. Bière qui est dans le top 10 des notes des utilisateurs du mois
   2. Bière dont le type est dans la bibliothèque de l’utilisateur
   3. Bière qui n’est pas dans la bibliothèque de l’utilisateur
2. L’Api Bières renvoie la liste de bières
3. L’Application Mobile affiche les résultats

*A3 :* l’utilisateur souhaite des suggestions « surprises »

1. L’Application Mobile fait appel à l’Api Bières pour obtenir une liste de 5 bières suggérées avec le critère :
   1. Bière aléatoirement sélectionnée
2. L’Api Bières renvoie la liste de bières
3. L’Application Mobile affiche les résultats

Enchainements d’erreurs :

*E1 :* Il n’existe pas de bière avec vues sponsorisées du même type que la bière préalablement sélectionnée

1. Aucune bière sponsorisée n’est affichée

*E2 :* Il n’existe pas de bière avec vues sponsorisées

1. Aucune bière sponsorisée n’est affichée

*E3 :* Moins de 5 bières répondent aux critères

1. L’Application Mobile affiche uniquement les bières renvoyées par l’Api Bières

*E4 :* Aucune bière ne répond aux critères

1. Une notification est affichée à l’utilisateur

### Etablissement/brasserie

#### Ecran



#### Description textuelle

Résumé : écran détaillant toutes les informations à propos d’un établissement. Il est précisé sur cet écran si l’établissement est ouvert, s’il s’agit d’un jour de fermeture exceptionnel ainsi que les horaires normaux d’ouvertures. Il permet à l’utilisateur d’aller voir la liste des bières que l’établissement vend.

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Etablissements *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile, a souhaité voir le détail d’un établissement

Scénario nominal :

1. L’Application Mobile fait appel à l’Api Etablissements pour obtenir les informations relatives à l’établissement
2. L’Api Etablissements renvoie toutes les informations de l’établissement
3. L’Application Mobile affiche les résultats
4. L’utilisateur clique sur le bouton « Voir la liste des bières »
5. L’Application Mobile renvoie l’utilisateur vers l’écran liste des bières - établissement

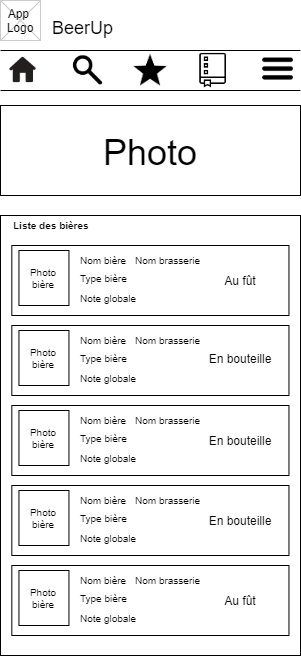
Scénarios alternatifs :

*A1 :* L’établissement ne possède pas de photo

1. Aucune photo n’est affichée

### Liste Bière Etablissement

#### Ecran



#### Description textuelle

Résumé : écran détaillant les bières vendues par un établissement. De plus, il est également affiché le type de service (si la bière est vendue au fût ou en bouteille).

Acteurs : Utilisateurs *(primaire),* l’Application Mobile *(secondaire),* l’Api Etablissements *(secondaire),* L’Api Bières *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile, a souhaité voir le détail des bières vendues par un établissement

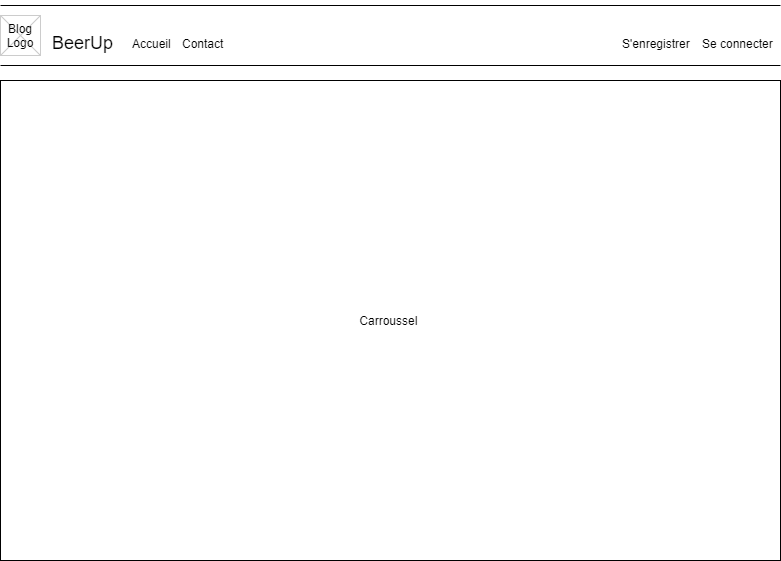
Scénario nominal :

1. L’Application Mobile fait appel à l’Api Etablissements pour obtenir la liste des bières servies par l’établissement ainsi que le type de service.
2. L’Api Etablissements renvoie la liste des bières et leurs types de service
3. L’Application Mobile fait appel à l’Api Bières pour obtenir le détail des bières servies par l’établissement.
4. L’Api Bières renvoie le détail des bières
5. L’Application Mobile affiche les résultats
6. L’utilisateur clique sur une bière dans la liste
7. L’Application Mobile renvoie l’utilisateur vers l’écran bière

## Site Web

### Accueil

#### Ecran



#### Description textuelle

Résumé : écran d’accueil avec un carrousel d’images au site web permettant à l’utilisateur de se connecter ou de s’enregistrer ou d’aller voir la page de contact.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web

Scénario nominal :

1. L’utilisateur clique sur « se connecter »
2. L’Application Web redirige l’utilisateur vers IdentityServer sur la page login

Scénarios alternatifs :

*A1 :* L’utilisateur clique sur la page contact

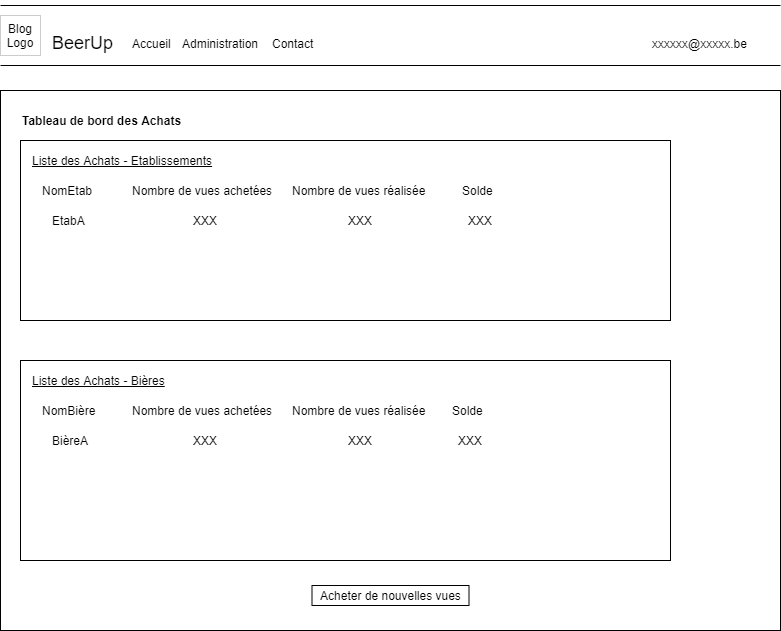
1. L’Application Web redirige l’utilisateur vers la page contact

*A2 :* L’utilisateur est déjà authentifié

1. Le menu « Administration » est visible à l’utilisateur et lui permet d’être redirigé vers les pages ad-hoc selon ses accès. (Voir règles d’autorisations et d’accès)

### Tableau de bord Achats

#### Ecran



#### Description textuelle

Résumé : écran tableau de bord des achats divisé en 2 parties, les achats de vues sponsorisées pour les établissements et celles pour les bières. Pour chaque bières et établissements possédés, le tableau affiche le nombre de vue achetées, le nombre de vue réalisées et le nombre de vue restantes. Il permet à l’utilisateur d’arriver sur la page d’achat de nouvelles vues sponsorisées.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Factures *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web et est authentifié
* L’utilisateur a comme rôle GroupAdmin ou GroupAchats

Scénario nominal :

1. L’Application Web fait appel à l’Api Factures pour obtenir la liste des achats pour les établissements possédés par l’organisation
2. L’Api Factures renvoie le tableau des données
3. L’Application Web fait appel à l’Api Factures pour obtenir la liste des achats pour les bières possédées par l’organisation
4. L’Api Factures renvoie le tableau des données
5. L’utilisateur clique sur « Acheter de nouvelles vues »
6. L’Application Web redirige l’utilisateur l’écran Achat

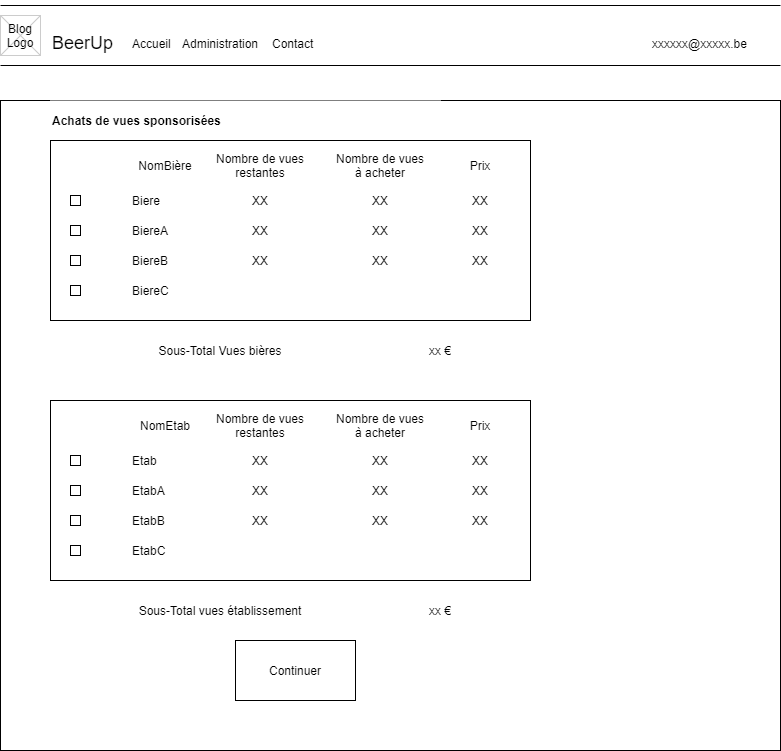
Scénarios alternatifs :

*A1 :* Il n’existe aucun établissement et/ou bières possédées par l’organisation

1. Les tableaux y relatifs sont laissés vides

### Achat

#### Ecran



#### Description textuelle

Résumé : écran des achats divisé en 2 parties, les achats de vues sponsorisées pour les établissements et celles pour les bières. Pour chaque bières et établissements possédés, le tableau affiche le nombre de vues restantes, un menu déroulant pour sélectionner le nombre de vues à acheter et le prix de cet achat. Les sous-totaux pour les établissements et les bières sont affichés en dessous des tableaux correspondants.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Factures *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web et est authentifié
* L’utilisateur a comme rôle GroupAdmin ou GroupAchats

Scénario nominal :

1. L’Application Web fait appel à l’Api Factures pour obtenir la liste des achats pour les établissements possédés par l’organisation
2. L’Api Factures renvoie le tableau des données
3. L’Application Web fait appel à l’Api Factures pour obtenir la liste des achats pour les bières possédées par l’organisation
4. L’Api Factures renvoie le tableau des données
5. L’Application Web fait appel à l’Api Factures pour obtenir les différents plans tarifaires pour les établissements
6. L’Api Factures renvoie le tableau des données
7. L’Application Web fait appel à l’Api Factures pour obtenir les différents plans tarifaires pour les bières
8. L’Api Factures renvoie le tableau des données
9. L’utilisateur sélectionne le nombre de vue qu’il souhaite acheter
10. L’Application Web affiche le prix pour le nombre de vue sélectionné
11. L’Application Web met à jour le sous total en dessous du tableau.
12. L’utilisateur appuie sur le bouton « continuer »
13. L’Application Web redirige l’utilisateur l’écran confirmation achats

Scénarios alternatifs :

*A1 :* Il n’existe aucun établissement et/ou bières possédées par l’organisation

1. Les tableaux y relatifs sont laissés vides

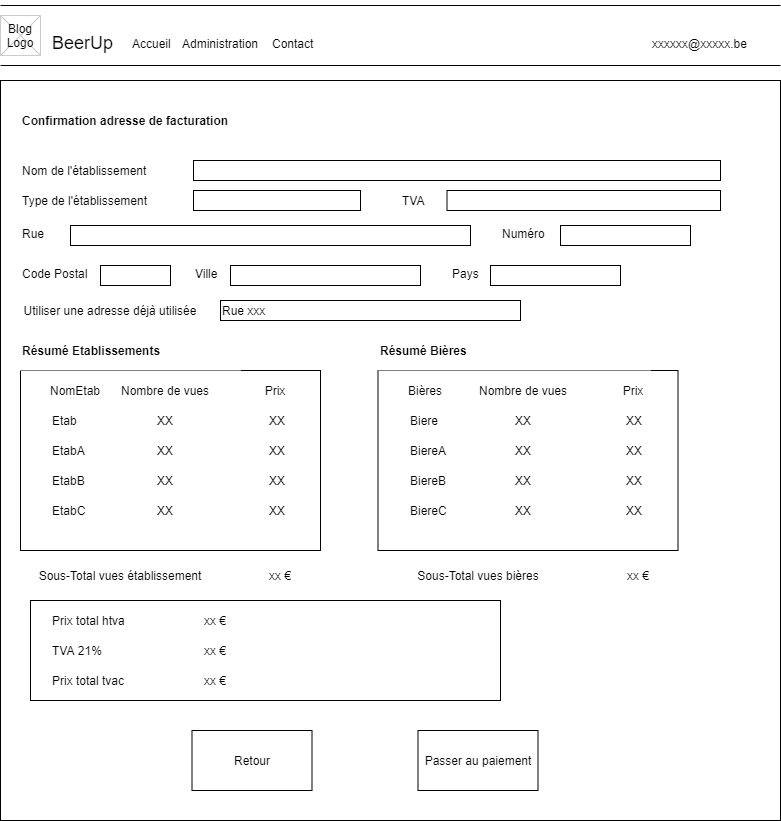
Enchainements d’erreurs :

*E1 :* L’utilisateur n’a rajouté aucun achat de vues et souhaite « continuer »

1. Un message d’erreur est affiché pour signifier qu’aucune vue n’a été rajoutée.

### Confirmation achats

#### Ecran



#### Description textuelle

Résumé : écran reprenant le résumé des achats souhaités par l’utilisateur, par catégorie et le total du montant à payer avec la TVA. Les champs adresses de facturation sont automatiquement préremplis avec la dernière adresse de facturation utilisée par l’organisation. Les champs nom, type et TVA ne sont pas modifiables par l’utilisateur. Une liste déroulante permet de sélectionner une adresse de facturation utilisée ultérieurement. L’écran permet également à l’utilisateur de soit revenir en arrière dans le choix des vues à acheter soit de passer au paiement.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Factures *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web et est authentifié
* L’utilisateur a comme rôle GroupAdmin ou GroupAchats

Scénario nominal :

1. L’Application Web fait appel à l’Api Factures pour obtenir les données concernant l’organisation de l’utilisateur
2. L’Api Factures renvoie les données
3. L’Application Web fait appel à l’Api Factures pour obtenir la liste des adresses de facturation pour l’organisation de l’utilisateur
4. L’Api Factures renvoie le tableau des données
5. L’Application Web affiche dans les champs la dernière adresse utilisée
6. L’Application Web affiche le champ Rue des adresses déjà utilisées dans la liste déroulante
7. L’Application Web affiche les différents achats souhaités par l’utilisateur
8. L’utilisateur clique sur passer au paiement.
9. L’Application Web redirige l’utilisateur vers le module de paiement.
10. Le module de paiement confirme que le paiement est effectué
11. L’Application Web envoie la commande à l’Api Facture pour enregistrement
12. L’Application Web redirige l’utilisateur l’écran factures

Scénarios alternatifs :

*A1 :* Il n’existe aucune adresse de facturation déjà utilisée

1. L’adresse préremplie est celle de l’organisation

*A2 :* L’utilisateur souhaite revenir au choix des vues à acheter

1. L’Application Web redirige l’utilisateur l’écran achat

Enchainements d’erreurs :

*E1 :* un champ d’adresse est laissé vide

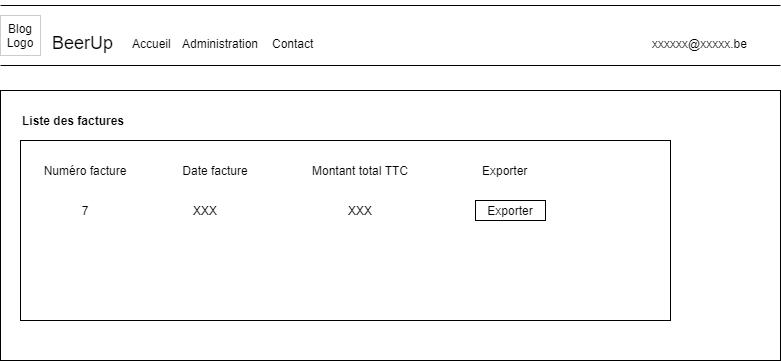
1. Un message d’erreur est affiché pour signifier que tous les champs doivent être remplis

*E2 :* le paiement n’a pas été effectué

1. Un message d’erreur est affiché pour signifier que le paiement a échoué

### Factures

#### Ecran



#### Description textuelle

Résumé : écran reprenant la liste des factures (numéro, date, montant total) pour l’organisation et permettant des les exporter individuellement en pdf. Les factures sont triées par ordre chronologique décroissant.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Factures *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web et est authentifié
* L’utilisateur a comme rôle GroupAdmin ou GroupAchats

Scénario nominal :

1. L’Application Web fait appel à l’Api Factures pour obtenir les données les factures de l’organisation
2. L’Api Factures renvoie les données
3. L’Application Web affiche sous forme de tableau les factures
4. L’utilisateur clique sur exporter pour une facture.
5. L’utilisateur reçoit un fichier pdf de la facture

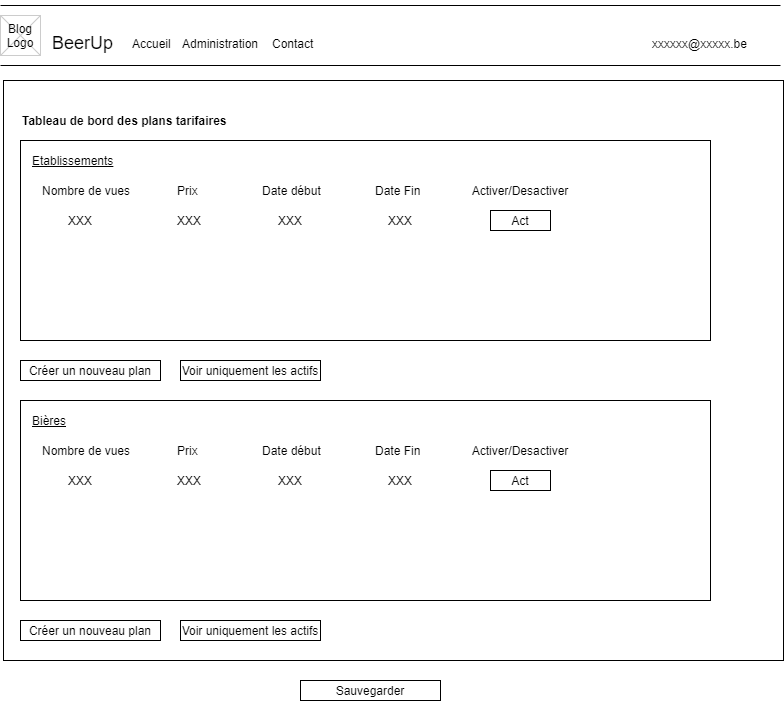
Scénarios alternatifs :

*A1 :* L’utilisateur souhaite changer le tri du tableau ou de l’ordre d’une colonne

1. L’utilisateur clique sur le titre d’une colonne
2. L’Application Web trie le tableau selon le critère choisi par l’utilisateur

### Tableau De Bord Plans tarifaires

#### Ecran



#### Description textuelle

Résumé : écran divisé en 2 parties : établissements et bières et reprenant l’entièreté des plans tarifaires actifs et non actifs et permettant à l’administrateur des les activer ou les désactiver pour les clients. L’écran permet également de filtrer les vues en affichant uniquement les tarifs actifs. L’administrateur peut également rajouter de nouveaux plans tarifaires ou encore modifier les dates d’activité.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Factures *(secondaire)*

Préconditions :

* L’utilisateur est allé sur l’Application Web et est authentifié
* L’utilisateur a comme rôle Administrateur

Scénario nominal :

1. L’Application Web fait appel à l’Api Factures pour obtenir les plans tarifaires
2. L’Api Factures renvoie les données
3. L’Application Web affiche sous forme de tableau les plans tarifaires
4. L’utilisateur clique pour rajouter un plan tarifaire
5. L’Application Web rajoute une ligne vierge au tableau ad-hoc et permet à l’utilisateur de remplir les différents champs
6. L’utilisateur rempli les différents champs
7. L’utilisateur clique sur le bouton « Sauvegarder »
8. L’Application Web envoie les tableaux à l’Api Factures pour sauvegarder les nouvelles lignes
9. L’Application Web recharge la page et affiche une notification à l’utilisateur que les modifications ont bien été enregistrées.

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite changer les dates d’un plan tarifaire

1. L’utilisateur modifie les champs date d’un plan tarifaire
2. L’utilisateur clique sur le bouton « Sauvegarder »
3. L’Application Web envoie les tableaux à l’Api Factures pour mettre à jour les lignes dont les dates ont été modifiées

*A2 :* L’utilisateur souhaite changer activer/désactiver un plan tarifaire

1. L’utilisateur clique sur le bouton activer/désactiver
2. L’Application Web modifie le bouton en désactiver/activer
3. L’utilisateur clique sur le bouton « Sauvegarder »
4. L’Application Web envoie les tableaux à l’Api Factures pour mettre à jour les lignes dont le statut actif a été modifié

*A3 :* L’utilisateur souhaite uniquement voir les plans tarifaires actifs

1. L’utilisateur clique sur le bouton « Voir uniquement les actifs »
2. L’Application Web filtre les plans tarifaires selon leur attribut actif et leur date de fin et de début.
3. L’Application Web modifie le bouton « Voir uniquement les actifs » en « Voir tout » et permet à l’utilisateur de réafficher l’entièreté des lignes

Enchainements d’erreurs :

*E1 :* un champ d’un plan tarifaire est laissé vide

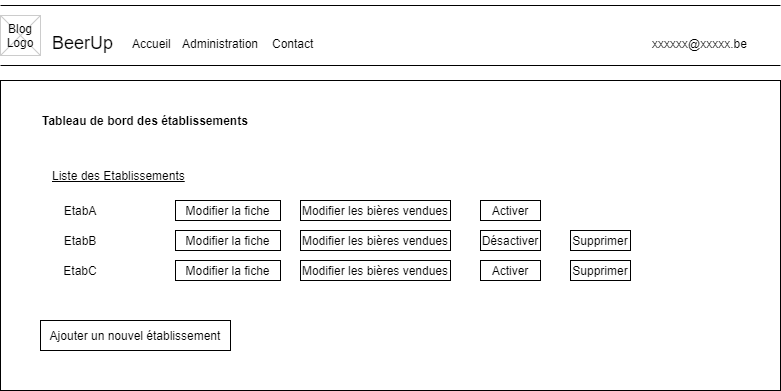
1. Un message d’erreur est affiché pour signifier que tous les champs doivent être remplis

*E2 :* une date de début est ultérieure à une date de fin

1. Un message d’erreur est affiché pour signifier que la date de début doit être antérieur à la date de fin.

### Tableau de bord Etablissement

#### Ecran



#### Description textuelle

Résumé : écran reprenant la liste des établissements de l’organisation pour lesquels l’utilisateur a accès et permettant à cet utilisateur de modifier selon ses accès la fiche de l’établissement, la liste des bières vendues de l’établissement, d’activer, de désactiver ou de supprimer l’établissement ou encore d’en ajouter un nouveau.

L’administrateur verra tous les établissements enregistrés au contraire des utilisateurs liés à une organisation qui seront limités à la vue des établissements liés à leur organisation.

Il est à noter que pour que le bouton Supprimer soit visible, aucune facture ne peut reprendre l’établissement en question ou encore être une brasserie possédant des bières

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Etablissements *(secondaire)*

Préconditions :

* L’Utilisateur est allé sur l’Application Web et est authentifié
* L’Utilisateur a comme rôle GroupAdmin ou GroupEtablissement ou GroupBiere

Scénario nominal :

1. L’Application Web fait appel à l’Api Etablissements pour obtenir les établissements liés à l’organisation
2. L’Api Etablissements renvoie les données
3. L’Application Web affiche sous forme de tableau les établissements
4. L’Utilisateur clique sur le bouton « modifier la fiche »
5. L’Application Web redirige l’utilisateur l’écran création/modification d’un établissement

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite modifier la liste de bières vendues par un établissement

1. L’utilisateur clique sur le bouton « modifier les bières vendues »
2. L’Application Web redirige l’utilisateur l’écran ajout/modification d’une bière

*A2 :* L’utilisateur souhaite ajouter un établissement

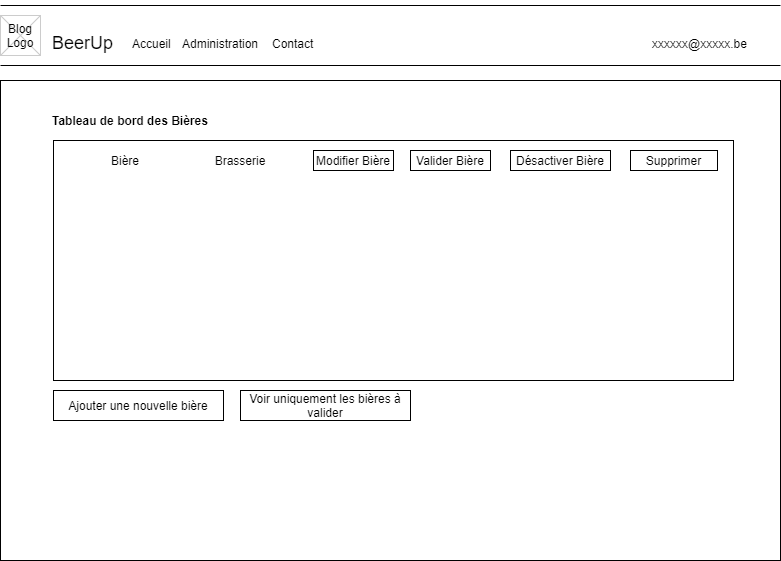
1. L’Utilisateur clique sur le bouton « ajouter un établissement »
2. L’Application Web redirige l’utilisateur l’écran création/modification d’un établissement

*A3 :* L’utilisateur souhaite supprimer un établissement

1. L’Utilisateur clique sur le bouton « supprimer »
2. L’Application Web affiche une notification pour demander la confirmation de la suppression.
3. L’Utilisateur confirme son choix
4. L’Application Web fait appel à l’Api Etablissements pour désactiver l’établissement
5. L’Api Etablissements confirme l’action
6. L’Application Web recharge la page et affiche une notification à l’utilisateur que les modifications ont bien été enregistrées.

### Tableau de bord des bières

#### Ecran



#### Description textuelle

Résumé : écran reprenant la liste des bières de l’organisation pour lesquelles l’utilisateur a accès et permettant à cet utilisateur de modifier selon ses accès la fiche de la bière, de valider la bière de désactiver la bière (elle ne sera plus visible sur l’Application Mobile) ou d’en ajouter une nouvelle. Une fonction permet d’afficher uniquement les bières à valider.

L’administrateur verra toutes les bières enregistrées au contraire des utilisateurs liés à une organisation qui seront limités à la vue des bières liées à leur organisation.

Il est à noter que les bières à valider sont celles qui ont été créées par un utilisateur de l’Application Mobile et assignées à une brasserie par cet utilisateur – l’action de les valider correspond à confirmer l’appartenance de la bière à la brasserie.

De plus, le bouton « supprimer » ne s’affiche que pour l’administrateur et pour les bières qui ne sont pas validées et qui ne sont pas reprises dans une facture.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Bières *(secondaire)*

Préconditions :

* L’Utilisateur est allé sur l’Application Web et est authentifié
* L’Utilisateur a comme rôle Administrateur ou GroupAdmin ou GroupBiere.

Scénario nominal :

1. L’Application Web fait appel à l’Api Bières pour obtenir les bières liées à l’organisation
2. L’Api Bières renvoie les données
3. L’Application Web affiche sous forme de tableau les établissements
4. L’Utilisateur clique sur le bouton « modifier la bière »
5. L’Application Web redirige l’utilisateur l’écran création/modification d’une bière

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite valider une bière (à inverser dans le cas de dévalider)

1. L’Utilisateur clique sur le bouton « valider »
2. L’Application Web fait appel à l’Api Bières pour valider la bière
3. L’Api Bières confirme la validation
4. L’Application Web modifie le bouton « valider » en « dévalider »

*A2 :* L’utilisateur souhaite activer une bière (à inverser dans le cas de désactiver)

1. L’Utilisateur clique sur le bouton « activer »
2. L’Application Web fait appel à l’Api Bières pour activer la bière
3. L’Api Bières confirme l’activation
4. L’Application Web modifie le bouton « activer » en « désactiver »

*A3 :* L’utilisateur souhaite ajouter une bière

1. L’Utilisateur clique sur le bouton « ajouter une nouvelle bière »
2. L’Application Web redirige l’utilisateur l’écran création/modification d’une bière

*A4 :* L’utilisateur est Administrateur et souhaite supprimer une bière

1. L’Utilisateur clique sur le bouton « supprimer »
2. L’Application Web affiche une notification pour demander la confirmation de la suppression.
3. L’Utilisateur confirme son choix
4. L’Application Web fait appel à l’Api Bières pour supprimer la bière
5. L’Api Bières confirme l’action
6. L’Application Web recharge la page et affiche une notification à l’utilisateur que la suppression a bien été enregistrée.

*A5 :* L’utilisateur souhaite afficher uniquement les bières à valider

1. L’Utilisateur clique sur le bouton « voir uniquement les bières à valider »
2. L’Application Web n’affiche plus que les bières qui ne sont pas valides.
3. L’application Web change le bouton « voir uniquement les bières à valider » en « voir tout »

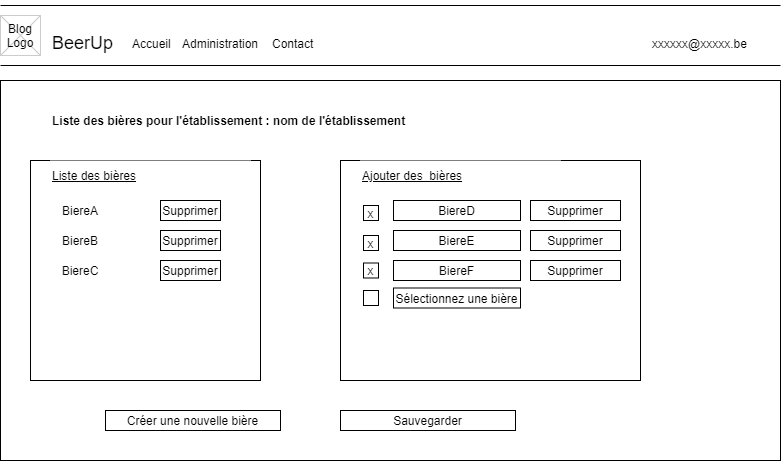
Enchainements d’erreurs :

*E1 :* L’utilisateur souhaite désactiver une bière qui n’est pas validée

1. Un message d’erreur est affiché pour signifier qu’on ne peut désactiver que les bières validées

### Ajout/modification de bières a un établissement

#### Ecran



#### Description textuelle

Résumé : écran permettant de supprimer ou de rajouter des bières vendues par un établissement.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Etablissements *(secondaire)*

Préconditions :

* L’Utilisateur est allé sur l’Application Web et est authentifié et est passé par le tableau de bord des établissements
* L’Utilisateur a comme rôle Administrateur ou GroupAdmin ou GroupBiere ou GroupEtablissement

Scénario nominal :

1. L’Application Web fait appel à l’Api Etablissements pour obtenir les bières liées à l’établissement
2. L’Api Etablissements renvoie les données
3. L’Application Web affiche sous forme de tableau les établissements
4. L’Utilisateur clique sélectionne une bière à rajouter dans le menu déroulant
5. L’Application Web coche la case devant le nom, fait apparaître un bouton « supprimer » à côté de la liste déroulante et rajoute une nouvelle liste déroulante en dessous de celle(s) existante(s)
6. L’Utilisateur clique sur le bouton « Sauvegarder »
7. L’Application Web envoie la liste des bières à rajouter à l’établissement à l’Api Etablissements
8. L’Api Etablissements confirme
9. L’Application Web rafraichi la page web

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite supprimer une bière de la liste des bières vendues par l’établissement

1. L’Utilisateur clique sur le bouton « supprimer » de la liste des bières existantes
2. L’Application Web demande une confirmation à l’Utilisateur
3. L’Utilisateur confirme
4. L’Application Web rajoute dans la liste des éléments à supprimer l’ID de la bière et cache la ligne
5. L’Utilisateur clique sur le bouton « Sauvegarder »
6. L’Application Web envoie la liste des bières à supprimer de l’établissement à l’Api Etablissements

*A2 :* L’utilisateur souhaite supprimer une bière qu’il avait rajouté à la liste des bières à rajouter.

1. L’Utilisateur clique sur le bouton « supprimer » de la liste des bières à rajouter
2. L’Application Web retire la ligne

*A3 :* L’utilisateur souhaite créer une nouvelle bière

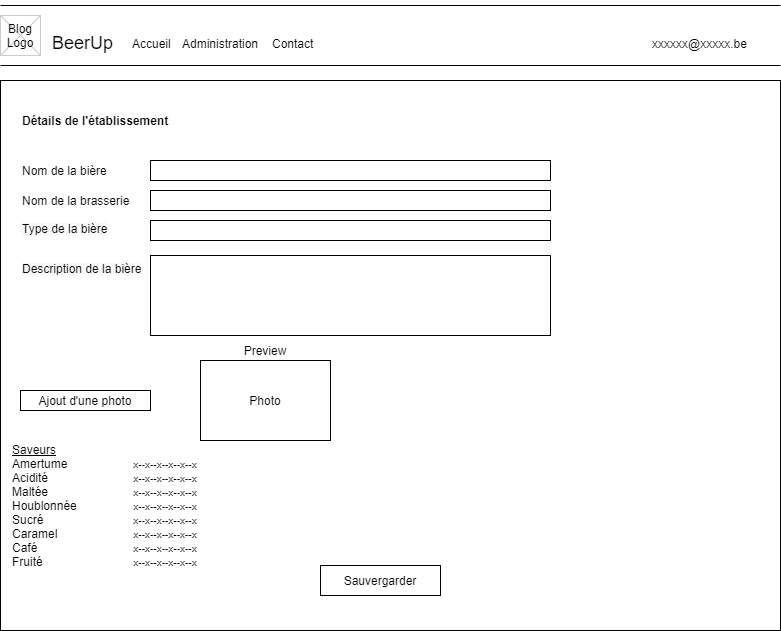
1. L’Utilisateur clique sur le bouton « créer une nouvelle bière »
2. L’Application Web redirige l’utilisateur vers l’écran création/modification d’une bière

*A4 :* L’utilisateur souhaite créer une nouvelle bière et a des bières qu’il souhaite rajouter et n’a pas sauvegardé

1. L’Utilisateur clique sur le bouton « créer une nouvelle bière »
2. L’Application Web notifie l’utilisateur qu’il risque de perdre ses modifications
3. L’Utilisateur confirme
4. L’Application Web redirige l’utilisateur vers l’écran création/modification d’une bière

### Création/Modification bière

#### Ecran



#### Description textuelle

Résumé : écran permettant d’ajouter ou de modifier une bière. Si la bière existe déjà, ses informations sont affichées. Le champ type de la bière est une liste déroulante

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Identity *(secondaire),* Api Bières *(secondaire)*

Préconditions :

* L’Utilisateur est allé sur l’Application Web et est authentifié et a souhaité créer ou modifier une bière
* L’Utilisateur a comme rôle Administrateur ou GroupAdmin ou GroupBiere avec le droit d’accès à cette bière

Scénario nominal :

1. L’Application Web fait appel à l’Api Bières pour obtenir les informations concernant la bière
2. L’Api Bières renvoie les données
3. L’Application Web affiche les données dans les champs modifiables par l’utilisateur
4. L’Application Web fait appel à l’Api Identity pour obtenir les Id de tous les utilisateurs de l’organisation de l’Utilisateur
5. L’Api Identity renvoie la liste des Id
6. L’Application Web demande l’avis actif lié à un de ces Id
7. L’Application Web affiche l’avis sur les saveurs.
8. L’utilisateur modifie les informations relatives à la bière
9. L’Application Web envoie l’Id de l’ancien avis à désactiver à l’Api Bière
10. L’Api Bières crée un nouvel avis et confirme l’action à l’Application Web
11. L’Application Web envoie l’objet Bière à l’Api Bière pour modification
12. L’Api Bières désactive l’avis et confirme l’action à l’Application Web
13. L’Application Web envoie le nouvel avis à l’Api Bière
14. L’Api Bières modifie la bière et confirme l’action à l’Application Web
15. L’Application Web renvoie l’utilisateur sur la page sur laquelle il se trouvait précédemment.
16. L’Application Web notifie l’utilisateur de la réussite de la modification

Scénarios alternatifs :

*A1 :* L’utilisateur crée une nouvelle bière

1. L’Utilisateur encode les informations relatives à la bière
2. L’Application Web renomme l’image et la sauvegarde sur le serveur en incluant l’Id de la bière dedans.
3. L’Application Web envoie l’objet Bière à l’Api Bière pour création
4. L’Api Bières vérifie si une brasserie de ce nom existe
   1. Si oui assigne l’ID de cette brasserie à la bière
   2. Si non crée une nouvelle brasserie et assigne l’ID à la bière
5. L’Api Bières crée la bière et confirme l’action à l’Application Web en renvoyant l’Id de la bière
6. L’Application Web envoie le nouvel avis à l’Api Bière
7. L’Api Bières crée un nouvel avis et confirme l’action à l’Application Web
8. L’Application Web renvoie l’utilisateur sur la page sur laquelle il se trouvait précédemment.
9. L’Application Web notifie l’utilisateur de la réussite de la création

*A2 :* L’utilisateur modifie la photo de la bière

1. L’Application Web renomme l’image et la sauvegarde sur le serveur en incluant l’Id de la bière dedans.
2. L’Application Web modifie la propriété image avec le nouveau nom dans l’objet Bière
3. L’Application Web supprime l’ancienne photo

Enchainements d’erreurs :

*E1 :* L’utilisateur n’a pas rempli tous les champs obligatoires (tous sauf la description)

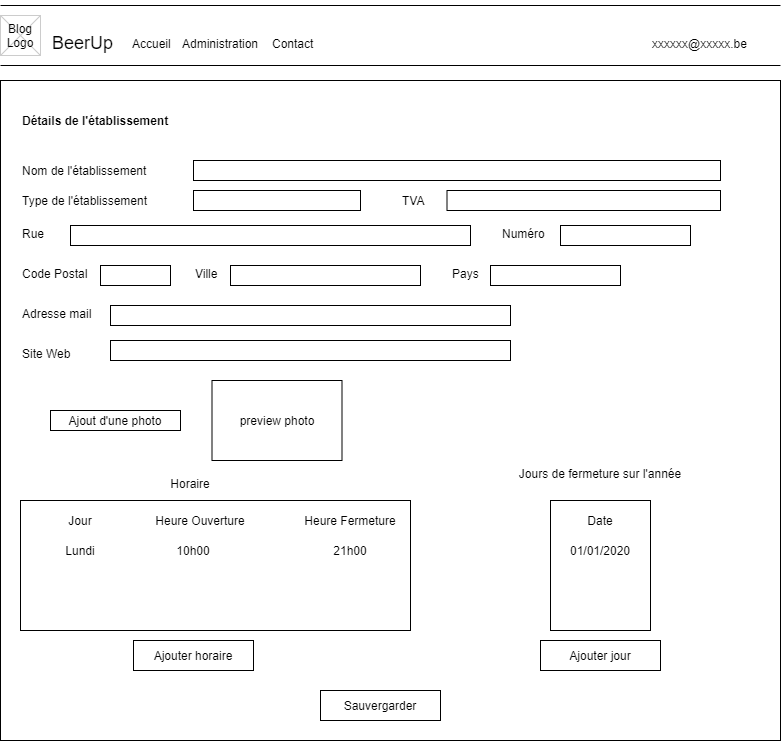
1. Un message d’erreur est affiché pour signifier le champ qu’il reste à remplir

*E2 :* L’utilisateur a déjà créé 50 bières aujourd’hui

1. Un message d’erreur est affiché pour signifier à l’utilisateur qu’il est limité à 50 bières par jour

### Création/Modification Etablissement

#### Ecran



#### Description textuelle

Résumé : écran permettant d’ajouter ou de modifier un établissement. Si l’établissement existe déjà, ses informations sont affichées. Le champ type de l’établissement est une liste déroulante.

Les horaires et les jours de fermetures sont triés chronologiquement. Seuls les jours de fermeture antérieurs à la date du jour sont affichés.

Acteurs : Utilisateurs *(primaire),* l’Application Web *(secondaire),* Api Etablissements *(secondaire)*

Préconditions :

* L’Utilisateur est allé sur l’Application Web et est authentifié et a souhaité créer ou modifier un établissement
* L’Utilisateur a comme rôle Administrateur ou GroupAdmin ou GroupEtablissement avec le droit d’accès à cet établissement

Scénario nominal :

1. L’Application Web fait appel à l’Api Etablissements pour obtenir les informations concernant l’établissement
2. L’Api Etablissements renvoie les données
3. L’Application Web affiche les données dans les champs modifiables par l’utilisateur
4. L’utilisateur modifie les informations relatives à la bière
5. L’Application Web envoie l’objet Etablissement à l’Api Etablissements pour modification
6. L’Api Etablissements modifie l’établissement et confirme l’action à l’Application Web
7. L’Application Web renvoie l’utilisateur sur la page tableau de bord des établissements
8. L’Application Web notifie l’utilisateur de la réussite de la modification

Scénarios alternatifs :

*A1 :* L’utilisateur crée un nouvel établissement

1. L’Utilisateur encode les informations relatives à l’établissement
2. L’Application Web envoie l’objet Etablissement à l’Api Etablissements pour création
3. L’Api Etablissement vérifie si le type de l’établissement est et si une brasserie de ce nom existe déjà mais sans autre renseignement que le nom
   1. Si oui assigne l’ID de cette brasserie à l’établissement
   2. Si non crée une nouvelle brasserie
4. L’Api Etablissement crée l’établissement et confirme l’action à l’Application Web en renvoyant l’Id de l’établissement
5. L’Application Web renomme l’image et la sauvegarde sur le serveur en incluant l’Id de l’établissement dedans.
6. L’Application Web renvoie l’utilisateur sur la page tableau de bord des établissements
7. L’Application Web notifie l’utilisateur de la réussite de la création

*A2 :* L’utilisateur modifie la photo de la bière

1. L’Application Web renomme l’image et la sauvegarde sur le serveur en incluant l’Id de la bière dedans.
2. L’Application Web modifie la propriété image avec le nouveau nom dans l’objet Bière
3. L’Application Web supprime l’ancienne photo

Enchainements d’erreurs :

*E1 :* L’utilisateur n’a pas rempli tous les champs obligatoires (nom, type, TVA, rue, numéro, code postal, ville, pays)

1. Un message d’erreur est affiché pour signifier le(s) champ(s) qu’il reste à remplir

*E2 :* Une brasserie du même nom existe déjà et comprend d’autres champs non-nuls que l’Id et le nom.

1. Un message d’erreur est affiché pour qu’une brasserie du même nom existe déjà

## IdentityServer

### Application Mobile

#### Login

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de se connecter via son login et mot de passe soit via Google ou Facebook. Il peut également créer un nouveau compte ou demander à renouveler son mot de passe.

Acteurs : Utilisateurs *(primaire)* et IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile

Scénario nominal :

1. L’utilisateur encode son login et son mot de passe
2. IdentityServer vérifie que les identifiants soient corrects
3. L’utilisateur est envoyé sur la page d’accueil

Scénarios alternatifs :

*A1 :* L’utilisateur se connecte via Facebook ou Google

1. IdentityServer vérifie que l’utilisateur existe bien
2. L’utilisateur est envoyé sur la page d’accueil

A2 : L’utilisateur se connecte via Facebook ou Google et se connecte pour la première fois

1. IdentityServer crée un nouvel utilisateur avec le token reçu du fournisseur externe.
2. L’utilisateur est envoyé sur la page d’accueil

A3 : L’utilisateur clique sur créer un compte

1. L’utilisateur est envoyé sur la page Register

A4 : L’utilisateur clique sur mot de passe oublié

1. Une notification est affichée pour annoncer qu’un email est envoyé avec le lien pour changer le mot de passe
2. Un email est envoyé avec un lien pour changer le mot de passe

Enchainements d’erreurs :

*E1 :* L’utilisateur encode de mauvais identifiants

1. Un message de validation s’affiche pour prévenir l’utilisateur

#### Register

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de créer un compte

Acteurs : Utilisateurs *(primaire)* et IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application mobile

Scénario nominal :

1. L’utilisateur encode son login, son mot de passe et le confirme.
2. IdentityServer vérifie que le mot de passe et que la confirmation de celui-ci soient identiques.
3. IdentityServer vérifie que l’adresse mail soit valide
4. IdentityServer vérifie que l’adresse mail ne soit pas déjà utilisée.
5. IdentityServer crée l’utilisateur dans la base de données
6. IdentityServer log l’utilisateur et crée le token
7. L’utilisateur est envoyé sur la page d’accueil

Enchainements d’erreurs :

*E1 :* L’utilisateur encode des mots de passe différents

1. Un message de validation s’affiche pour prévenir l’utilisateur

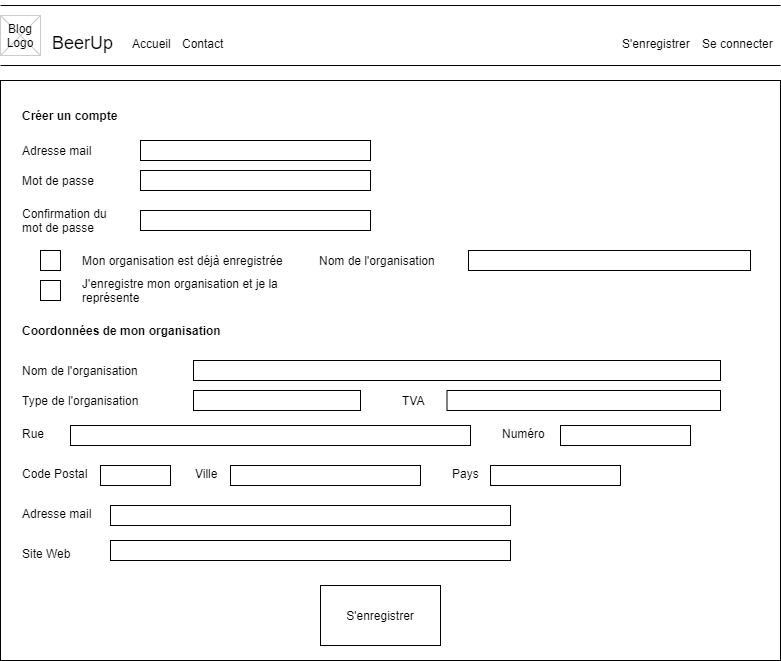
*E2 :* L’utilisateur encode une adresse mail non valide ou déjà utilisée

1. Un message de validation s’affiche pour prévenir l’utilisateur

### Application Web

#### Register

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de créer un compte et de renseigner l’organisation à laquelle il appartient.

L’utilisateur doit cocher soit l’option que son organisation existe déjà et alors une liste déroulante apparait pour lui permettre de sélectionner son organisation soit il enregistre également son organisation et les champs d’enregistrement de l’organisation apparaissent alors.

Lorsque l’Utilisateur coche une case, l’autre est automatiquement décochée.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire),* Api Organisations *(secondaire),* Api VatLayer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité s’enregistrer

Scénario nominal :

1. IdentityServer fait appel à l’Api Etablissements pour obtenir la liste des organisations
2. L’Api Organisations renvoie la liste des organisations
3. IdentityServer affiche la liste déroulante avec les organisations existantes
4. L’Utilisateur encode son login, son mot de passe et le champ de confirmation.
5. L’Utilisateur coche la case « J’enregistre mon organisation et je la représente »
6. IdentityServer affiche les champs d’enregistrement de l’organisation
7. L’Utilisateur rempli tous les champs
8. L’Utilisateur clique sur « s’enregistrer »
9. IdentityServer vérifie que tous les champs obligatoires soient remplis et valides
10. IdentityServer envoie le numéro de TVA à l’Api VATLayer pour vérification
11. L’Api VATLayer confirme que le numéro est valide
12. IdentityServer crée l’utilisateur
13. IdentityServer log l’utilisateur et crée le token
14. IdentityServer envoie l’organisation à l’Api Organisations pour enregistrement
15. L’Api Etablissements enregistre l’organisation et renvoie l’Id assignée à celle-ci
16. IdentityServer assigne l’Id de l’organisation à l’Utilisateur
17. IdentityServer assigne le rôle de GroupAdmin à l’Utilisateur
18. L’utilisateur est envoyé sur la page d’accueil de l’Application Web

Scénarios alternatifs :

*A1 :* L’organisation de l’Utilisateur est déjà enregistrée

1. L’Utilisateur coche la case « Mon organisation est déjà enregistrée »
2. L’utilisateur sélectionne son organisation
3. L’Utilisateur clique sur « s’enregistrer »
4. IdentityServer vérifie que l’adresse mail ne soit pas déjà utilisée.
5. IdentityServer crée l’utilisateur dans la base de données
6. IdentityServer log l’utilisateur et crée le token
7. L’utilisateur est envoyé sur la page d’accueil de l’Application Web

Enchainements d’erreurs :

*E1 :* L’utilisateur encode des mots de passe différents

1. Un message de validation s’affiche pour prévenir l’utilisateur

*E2 :* L’utilisateur encode une adresse mail non valide ou déjà utilisée

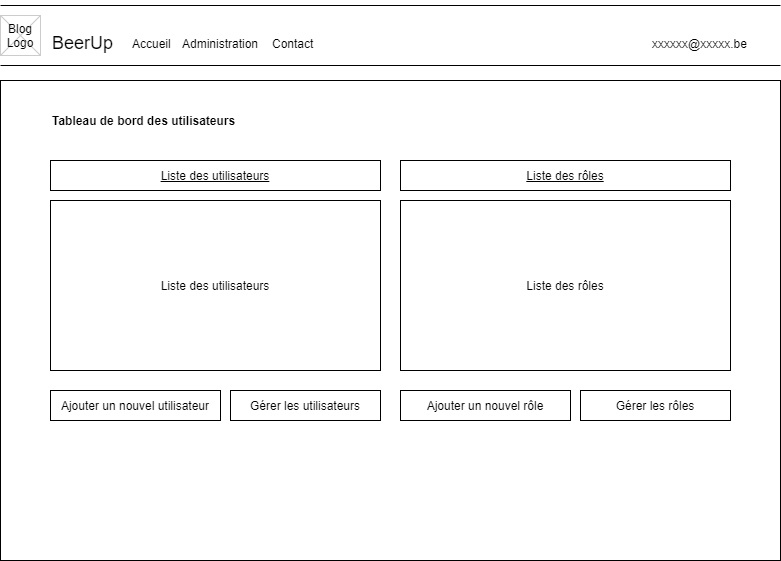
1. Un message de validation s’affiche pour prévenir l’utilisateur

*E3 :* Une organisation avec le même nom et numéro de tva existe déjà

1. Un message d’erreur s’affiche pour prévenir l’utilisateur

#### Tableau de bord utilisateur

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les utilisateurs de l’organisation ainsi que les rôles et de naviguer entre les différentes options de gestion des utilisateurs. A noter qu’uniquement l’Administrateur voit le bouton d’ajout d’un nouveau rôle.

Si des utilisateurs liés à l’organisation ne sont pas encore validés, une notification est affichée

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité aller sur le tableau de bord des Utilisateurs
* L’utilisateur est soit Administrateur soit GroupAdmin

Scénario nominal :

1. IdentityServer vérifie s’il existe des utilisateurs non validés et liés à l’organisation
2. IdentityServer affiche tous les utilisateurs liés à l’organisation
3. IdentityServer affiche tous les rôles
4. L’utilisateur clique sur le bouton « Gérer les Utilisateurs »
5. IdentityServer redirige l’utilisateur sur la page « Gestion utilisateur »

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite créer un nouvel utilisateur

1. L’utilisateur clique sur le bouton « Ajouter un nouvel utilisateur »
2. IdentityServer redirige l’utilisateur sur la page « Register »

*A2 :* L’utilisateur souhaite gérer les rôles

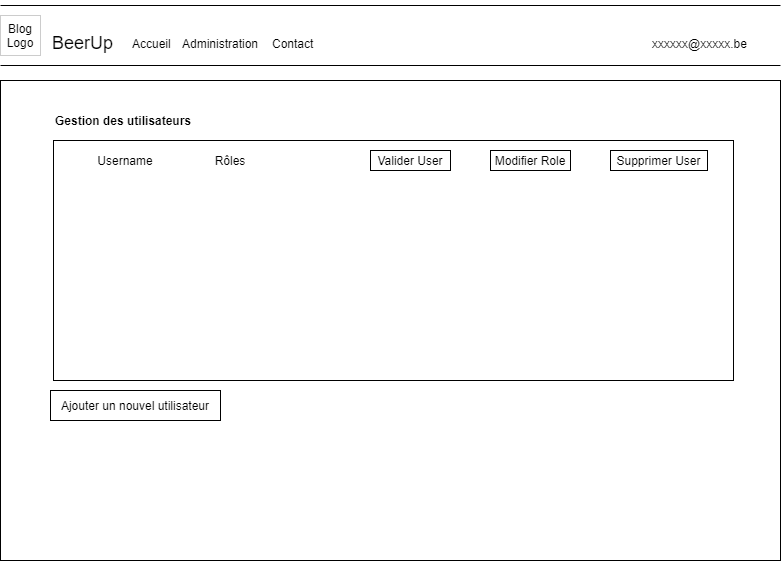
1. L’utilisateur clique sur le bouton « gérer les rôles »
2. IdentityServer redirige l’utilisateur sur la page « Gestion des rôles »

*A3 :* L’utilisateur est Administrateur et souhaite créer un nouveau rôle

1. L’utilisateur clique sur le bouton « Créer un nouveau Rôle »
2. IdentityServer redirige l’utilisateur sur la page « Register »

#### Gestion Utilisateurs

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les utilisateurs de l’organisation avec leurs rôles. Il permet également de valider et de supprimer un utilisateur ainsi que de modifier les rôles d’un utilisateur.

L’écran permet également la redirection vers la création d’un nouvel utilisateur.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité gérer ses utilisateurs
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche pour chaque utilisateur lié à l’organisation l’username, la liste des rôles ainsi que des boutons pour valider/dévalider un utilisateur, modifier les rôles d’un utilisateur ou supprimer un utilisateur
2. L’utilisateur clique sur modifier le rôle
3. IdentityServer redirige l’utilisateur sur la page « Modification Rôles d’un GroupUser »

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite créer un nouvel utilisateur

1. L’utilisateur clique sur le bouton « Ajouter un nouvel utilisateur »
2. IdentityServer redirige l’utilisateur sur la page « Register »

*A2 :* L’utilisateur souhaite valider un utilisateur (à inverser pour dévalider)

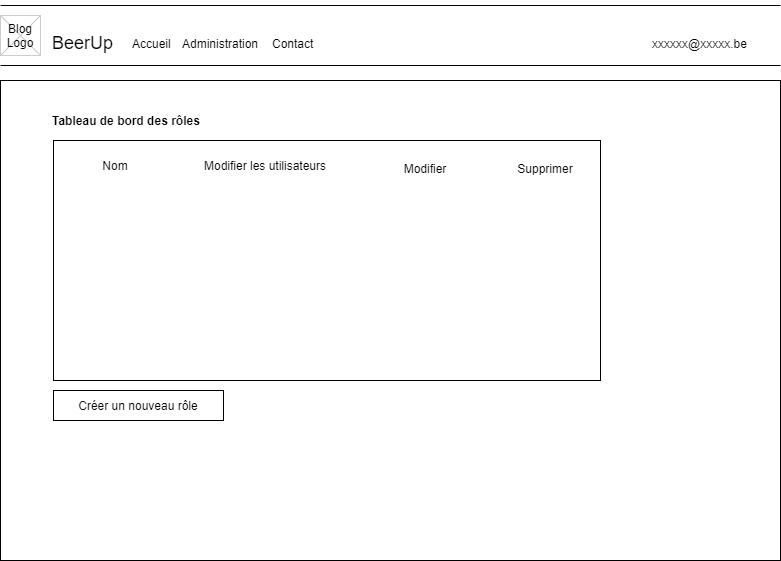
1. L’utilisateur clique sur le bouton « valider »
2. IdentityServer valide l’utilisateur
3. IdentityServer recharge la page
4. Le bouton « valider » est devenu « dévalider »

*A3 :* L’utilisateur souhaite supprimer un utilisateur

1. L’utilisateur clique sur le bouton « supprimer »
2. IdentityServer demande confirmation à l’utilisateur
3. L’utilisateur confirme
4. IdentityServer supprime l’utilisateur
5. IdentityServer recharge la page

#### Gestion des rôles

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents rôles. De plus un bouton permet de modifier les utilisateurs auxquels le rôle a été attribué. De plus l’Administrateur a 3 boutons supplémentaires qui lui permettent de créer, modifier et supprimer un rôle.

L’écran permet également la redirection vers la création d’un nouvel utilisateur.

Le rôle d’Administrateur n’est visible que pour un Administrateur

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité gérer les rôles
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des rôles
2. L’utilisateur clique sur modifier les utilisateurs
3. IdentityServer redirige l’utilisateur sur la page « Modification des utilisateurs ayant un rôle »

Scénarios alternatifs :

*A1 :* L’utilisateur est Administrateur et souhaite créer un nouveau rôle

1. L’utilisateur clique sur le bouton « Créer un nouveau rôle »
2. IdentityServer redirige l’utilisateur sur la page « Créer un nouveau rôle »

*A2 :* L’utilisateur est Administrateur et modifier un rôle

1. L’utilisateur clique sur le bouton « modifier »
2. IdentityServer redirige l’utilisateur sur la page « Modifier un rôle »

*A3 :* L’utilisateur est Administrateur et souhaite supprimer un rôle

1. L’utilisateur clique sur le bouton « supprimer »
2. IdentityServer demande confirmation à l’utilisateur
3. L’utilisateur confirme
4. IdentityServer supprime le rôle
5. IdentityServer recharge la page

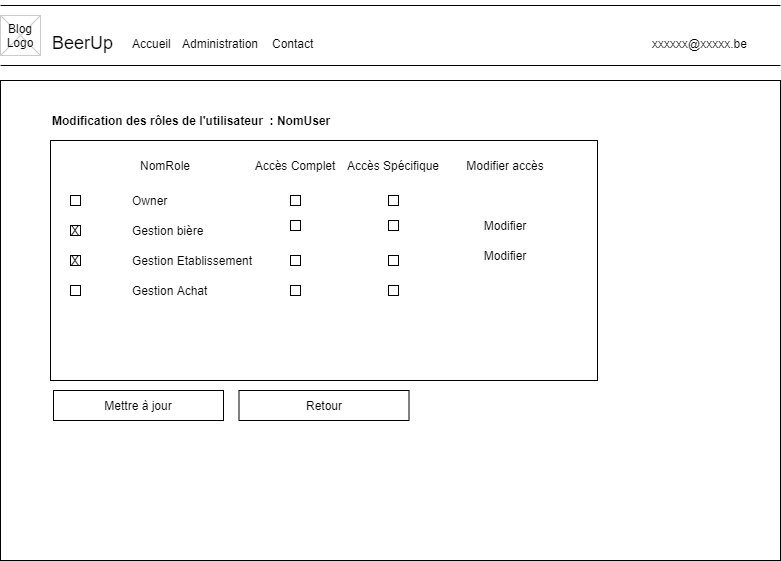
Enchainements d’erreurs :

*E1 :* Des utilisateurs possèdent toujours le rôle que l’utilisateur souhaite supprimer

1. Un message de notification s’affiche pour prévenir l’utilisateur que seuls les rôles sans utilisateurs peuvent être supprimés.

#### Modification Rôles d’un GroupUser

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents rôles d’un utilisateur. Cocher ou découcher la case devant le nom d’un rôle permet de donner ou de retirer le rôle à l’utilisateur. Pour les rôles déjà existants un bouton modifier est également affiché permettant de personnaliser l’accès de l’utilisateur, par défaut le rôle donne un accès complet à l’objet du rôle.

Le rôle d’Administrateur n’est visible que pour un Administrateur

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité gérer les rôles d’un utilisateur
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des rôles, coche les cases pour lesquels l’utilisateur possède le rôle et affiche un bouton modifier
2. L’utilisateur rajoute un rôle à l’utilisateur en cochant la case adéquate
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer affecte le rôle à l’utilisateur
5. IdentityServer affecte le claim correspondant au rôle avec la valeur « all »
6. IdentityServer recharge la page

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite retirer un rôle à l’utilisateur

1. L’utilisateur retire un rôle à l’utilisateur en décochant la case adéquate
2. L’utilisateur appuie sur le bouton « mettre à jour »
3. IdentityServer retire le rôle à l’utilisateur
4. IdentityServer retire tous les claims de l’utilisateur correspondant au rôle
5. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite modifier le rôle GroupAchat affecté à l’utilisateur

1. L’utilisateur clique sur le bouton « modifier »
2. IdentityServer redirige l’utilisateur sur la page « Modification Accès GroupAchat »

*A3 :* L’utilisateur souhaite modifier le rôle GroupBière affecté à l’utilisateur

1. L’utilisateur clique sur le bouton « modifier »
2. IdentityServer redirige l’utilisateur sur la page « Modification Accès GroupBière»

*A4 :* L’utilisateur souhaite modifier le rôle GroupEtablissement affecté à l’utilisateur

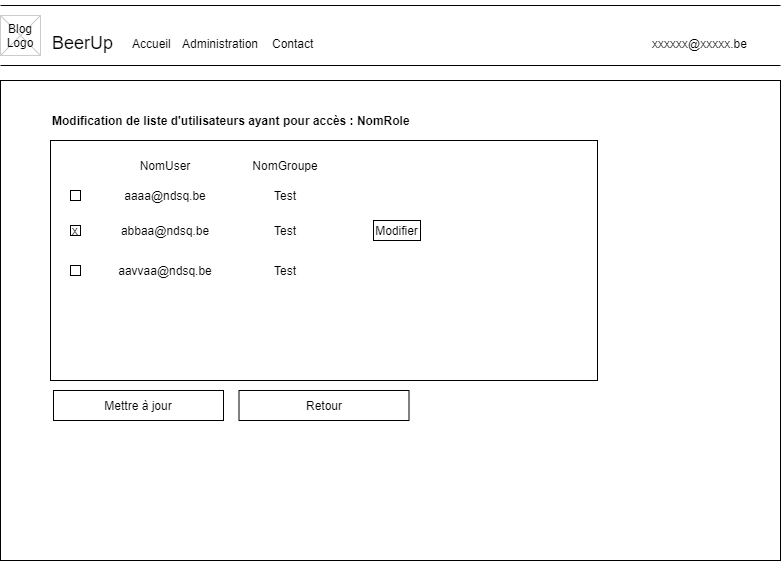
1. L’utilisateur clique sur le bouton « modifier »
2. IdentityServer redirige l’utilisateur sur la page « Modification Accès GroupEtablissement»

*A5 :* L’utilisateur revenir à la page précédente

1. L’utilisateur clique sur le bouton « retour »
2. IdentityServer redirige l’utilisateur sur la page précédente

#### Modification des utilisateurs ayant un rôle

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents utilisateurs d’un rôle. Cocher ou découcher la case devant le nom d’un utilisateur permet de lui donner ou de lui retirer le rôle. Pour les utilisateurs ayant le rôle, un bouton modifier est également affiché permettant de personnaliser l’accès de l’utilisateur, par défaut le rôle donne un accès complet à l’objet du rôle.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité gérer les utilisateurs d’un rôle
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des utilisateurs, coche les cases pour lesquels l’utilisateur possède le rôle et affiche un bouton modifier
2. L’utilisateur rajoute le rôle à l’utilisateur en cochant la case adéquate
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer affecte le rôle à l’utilisateur
5. IdentityServer affecte le claim correspondant au rôle avec la valeur « all »
6. IdentityServer recharge la page

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite retirer un rôle à l’utilisateur

1. L’utilisateur retire un rôle à l’utilisateur en décochant la case adéquate
2. L’utilisateur appuie sur le bouton « mettre à jour »
3. IdentityServer retire le rôle à l’utilisateur
4. IdentityServer retire tous les claims de l’utilisateur correspondant au rôle
5. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite modifier le rôle GroupAchat affecté à l’utilisateur (selon le rôle sélectionné)

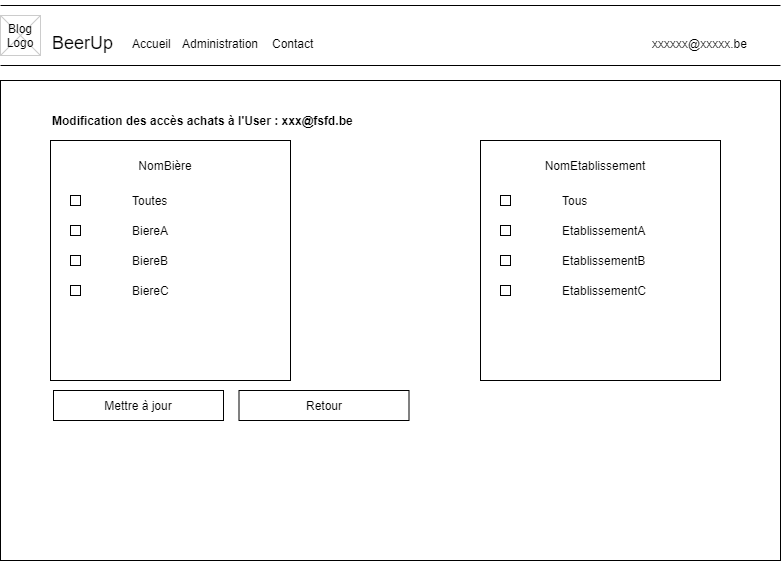
1. L’utilisateur clique sur le bouton « modifier »
2. IdentityServer redirige l’utilisateur sur la page « Modification Accès GroupAchat »

*A3 :* L’utilisateur revenir à la page précédente

1. L’utilisateur clique sur le bouton « retour »
2. IdentityServer redirige l’utilisateur sur la page précédente

#### Modification Accès GroupAchat

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents claims d’un utilisateur pour le rôle GroupAchat.

Cocher une case Toutes/Tous décoche automatiquement les autres de la liste en question. Cocher une autre case décoche la case Toutes/Tous.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité modifier les claims d’un utilisateur pour un rôle
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des bières et des établissements, coche les cases pour lesquels l’utilisateur possède un claim
2. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer affecte le claim à l’utilisateur
5. IdentityServer recharge la page

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite retirer un claim à l’utilisateur

1. L’utilisateur retire un claim à l’utilisateur en décochant la case adéquate
2. L’utilisateur appuie sur le bouton « mettre à jour »
3. IdentityServer retire le claim à l’utilisateur
4. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim « all » par un claim spécifique

1. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
2. IdentityServer décoche la case Tous/Toutes
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire le claim All à l’utilisateur
5. IdentityServer affecte le claim spécifique à l’utilisateur
6. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim spécifique par un claim « all »

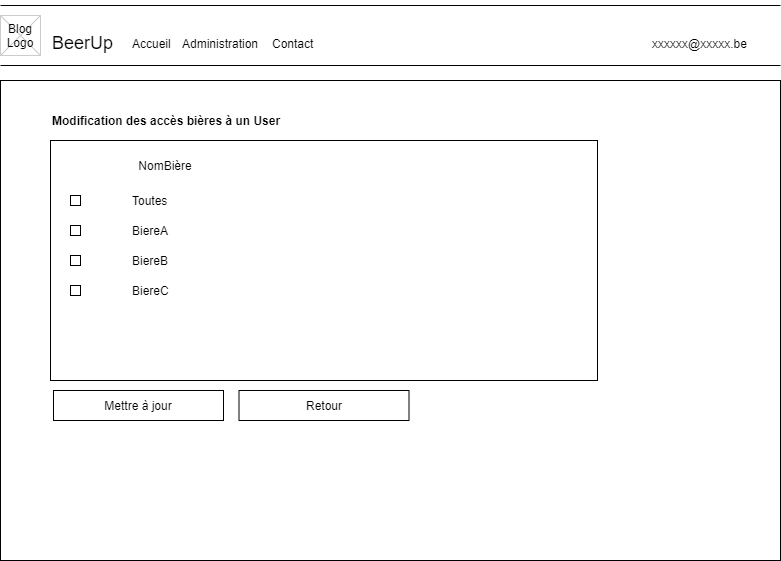
1. L’utilisateur coche la case Tous/Toutes
2. IdentityServer décoche les autres cases de la liste
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire tous les claims spécifiques à l’utilisateur
5. IdentityServer affecte le claim All à l’utilisateur
6. IdentityServer recharge la page

*A4 :* L’utilisateur revenir à la page précédente

1. L’utilisateur clique sur le bouton « retour »
2. IdentityServer redirige l’utilisateur sur la page précédente

#### Modification Accès GroupBière

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents claims d’un utilisateur pour le rôle GroupBière.

Cocher la case Toutes décoche automatiquement les autres de la liste en question. Cocher une autre case décoche la case Toutes.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité modifier les claims d’un utilisateur pour un rôle
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des bières, coche les cases pour lesquels l’utilisateur possède un claim
2. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer affecte le claim à l’utilisateur
5. IdentityServer recharge la page

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite retirer un claim à l’utilisateur

1. L’utilisateur retire un claim à l’utilisateur en décochant la case adéquate
2. L’utilisateur appuie sur le bouton « mettre à jour »
3. IdentityServer retire le claim à l’utilisateur
4. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim « all » par un claim spécifique

1. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
2. IdentityServer décoche la case Toutes
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire le claim All à l’utilisateur
5. IdentityServer affecte le claim spécifique à l’utilisateur
6. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim spécifique par un claim « all »

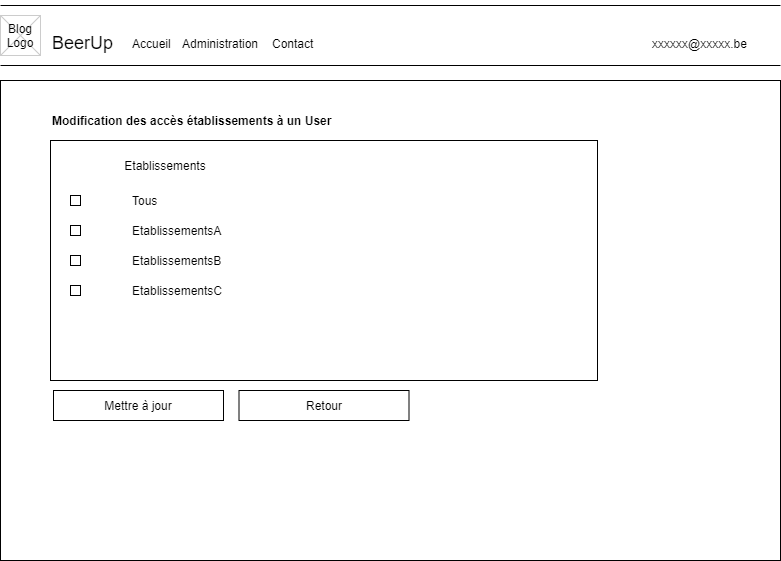
1. L’utilisateur coche la case Toutes
2. IdentityServer décoche les autres cases de la liste
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire tous les claims spécifiques à l’utilisateur
5. IdentityServer affecte le claim All à l’utilisateur
6. IdentityServer recharge la page

*A4 :* L’utilisateur revenir à la page précédente

1. L’utilisateur clique sur le bouton « retour »
2. IdentityServer redirige l’utilisateur sur la page précédente

#### Modification Accès GroupEtablissement

##### Ecran



##### Description textuelle

Résumé : permet à l’utilisateur de visualiser les différents claims d’un utilisateur pour le rôle GroupEtablissement.

Cocher la case Tous décoche automatiquement les autres de la liste en question. Cocher une autre case décoche la case Tous.

Acteurs : Utilisateurs *(primaire)*, IdentityServer *(secondaire)*

Préconditions :

* L’utilisateur a lancé l’application web et a souhaité modifier les claims d’un utilisateur pour un rôle
* L’utilisateur est soit Administrateur soit GroupOwner

Scénario nominal :

1. IdentityServer affiche la liste des établissements, coche les cases pour lesquels l’utilisateur possède un claim
2. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer affecte le claim à l’utilisateur
5. IdentityServer recharge la page

Scénarios alternatifs :

*A1 :* L’utilisateur souhaite retirer un claim à l’utilisateur

1. L’utilisateur retire un claim à l’utilisateur en décochant la case adéquate
2. L’utilisateur appuie sur le bouton « mettre à jour »
3. IdentityServer retire le claim à l’utilisateur
4. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim « all » par un claim spécifique

1. L’utilisateur rajoute un claim à l’utilisateur en cochant la case adéquate
2. IdentityServer décoche la case Tous
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire le claim All à l’utilisateur
5. IdentityServer affecte le claim spécifique à l’utilisateur
6. IdentityServer recharge la page

*A2 :* L’utilisateur souhaite changer le claim spécifique par un claim « all »

1. L’utilisateur coche la case Tous
2. IdentityServer décoche les autres cases de la liste
3. L’utilisateur appuie sur le bouton « mettre à jour »
4. IdentityServer retire tous les claims spécifiques à l’utilisateur
5. IdentityServer affecte le claim All à l’utilisateur
6. IdentityServer recharge la page

*A4 :* L’utilisateur revenir à la page précédente

1. L’utilisateur clique sur le bouton « retour »
2. IdentityServer redirige l’utilisateur sur la page précédente

# Contraintes fonctionnelles

## Règles d’accès et autorisations

### Application mobile

#### Utilisateur

|  |  |
| --- | --- |
| BU-RAA001 | Un utilisateur doit être soit enregistré soit se connecter via Google ou Facebook. |
| BU-RAA002 | Un utilisateur ne peut rajouter que 5 nouvelles bières par jour |

### Site Web

#### Administrateur

|  |  |
| --- | --- |
| BU-RAA003 | L’administrateur peut créer/modifier/supprimer tous les clients et utilisateurs |
| BU-RAA004 | L’administrateur peut créer un rôle |
| BU-RAA005 | L’administrateur peut modifier/supprimer les rôles de tous les clients |
| BU-RAA006 | L’administrateur peut créer/modifier tous les établissements |
|  | L’administrateur peut supprimer tous les établissements qui ne sont pas repris dans une facture et qui ne sont pas une brasserie possédant des bières |
|  | L’administrateur peut activer/désactiver tous les établissements |
| BU-RAA007 | L’administrateur peut créer/modifier toutes les brasseries |
|  | L’administrateur peut supprimer toutes les brasseries qui ne sont pas repris dans une facture |
|  | L’administrateur peut activer/désactiver toutes les brasseries |
| BU-RAA008 | L’administrateur peut créer/modifier/ valider toutes les bières |
|  | L’administrateur peut supprimer toutes les bières qui ne sont pas repris dans une facture et qui |
|  | L’administrateur peut activer/désactiver toutes les bières |
| BU-RAA009 | L’administrateur peut visualiser toutes les ventes |

#### GroupAdmin

|  |  |
| --- | --- |
| BU-RAA010 | Le premier client à s’enregistrer pour un groupe devient le GroupAdmin pour le groupe. |
| BU-RAA011 | Le GroupAdmin peut créer/modifier/supprimer tous les clients pour son groupe |
| BU-RAA012 | Le GroupAdmin peut modifier/supprimer les rôles de tous ses GroupUsers |
| BU-RAA013 | Le GroupAdmin peut créer/modifier tous les établissements pour son groupe |
|  | Le GroupAdmin peut supprimer tous les établissements qui ne sont pas repris dans une facture |
|  | Le GroupAdmin peut activer/désactiver tous les établissements qui sont repris dans une facture |
| BU-RAA014 | Le GroupAdmin peut créer/modifier toutes les brasseries pour son groupe |
|  | Le GroupAdmin peut supprimer toutes les brasseries qui ne sont pas repris dans une facture |
|  | Le GroupAdmin peut activer/désactiver toutes les brasseries qui sont repris dans une facture |
| BU-RAA015 | Le GroupAdmin peut créer/modifier/valider/activer/désactiver toutes les bières pour son groupe |
| BU-RAA016 | Le GroupAdmin peut rajouter jusqu’à 50 bières par jour |
| BU-RAA017 | Le GroupAdmin peut visualiser et faire de nouveaux achats de vues |

#### GroupBiere

|  |  |
| --- | --- |
| BU-RAA018 | Le GroupBière peut créer/modifier/valider/activer/désactiver pour une ou plusieurs bières pour son groupe |
| BU-RAA019 | Le GroupBière peut rajouter jusqu’à 50 bières par jour |

#### GroupAchat

|  |  |
| --- | --- |
| BU-RAA020 | Le GroupAchat peut visualiser et faire de nouveaux achats de vues pour tout le groupe |

#### GroupEtablissement

|  |  |
| --- | --- |
| BU-RAA021 | Le GroupEtablissement peut créer/modifier un ou plusieurs établissements pour son groupe |

#### GroupBrasserie

|  |  |
| --- | --- |
| BU-RAA022 | Le GroupBrasserie peut créer/modifier toutes les brasseries pour son groupe |

#### GroupUser

|  |  |
| --- | --- |
| BU-RAA023 | Le GroupUser est lié à un groupe |
| BU-RAA024 | Le GroupUser peut avoir plusieurs rôles |
| BU-RAA025 | Le GroupUser doit être validé par son GroupAdmin pour avoir le moindre accès. |

## Règles de structure

### Bière

|  |  |
| --- | --- |
| BU-RS001 | Une bière possède un nom |
| BU-RS002 | Une bière possède un brasseur |
| BU-RS003 | Une bière peut posséder une description de 1000 caractères maximum |
| BU-RS004 | Une bière possède un type |
| BU-RS005 | Une bière a un ou plusieurs avis |
| BU-RS006 | Une bière est ajoutée par un utilisateur ou un client |
| BU-RS007 | Une bière peut être vendue dans un ou plusieurs établissements |
| BU-RS008 | Une bière peut être valide |

### TypeBière

|  |  |
| --- | --- |
| BU -RS009 | Il existe 8 types : blonde, brune, triple, blanche, noire, gueuze, rouge, fruitée |

### AvisBière

|  |  |
| --- | --- |
| BU-RS010 | Un avis est réalisé par un utilisateur ou par un groupe |
| BU-RS011 | Un avis concerne une bière |
| BU-RS012 | Un avis possède 8 gouts différents : Amertume, Houblonné, Caramel, Café, Malté, Sucré, Acide, Fruité |
| BU-RS013 | Chaque goût est noté de 0 à 5 et ne peut être nul |
| BU-RS014 | Une bière possède une note générale |
| BU-RS015 | Chaque note générale est évaluée de 0 à 5 et peut être nul si l’avis est réalisé par un groupe |

### Organisation

|  |  |
| --- | --- |
| BU-RS016 | Une organisation possède un nom, un type, un numéro de tva, une rue, un numéro de rue, un code postal, une ville, un pays. |
| BU-RS017 | Une organisation peut posséder une adresse mail et un site web. |

### TypeOrganisation

|  |  |
| --- | --- |
| BU-RS018 | Il existe 3 types : producteur, vendeur, hybride |

### Facture

|  |  |
| --- | --- |
| BU-RS019 | Une facture possède une adresse de facturation |
| BU-RS020 | Une facture possède un ou plusieurs achats |
| BU-RS021 | Une facture est datée |
| BU-RS022 | Une facture possède un taux de tva |
| BU-RS023 | Une facture est liée à une organisation |
| BU-RS024 | Une facture possède un numéro unique et incrémental |

### Adressefacturation

|  |  |
| --- | --- |
| BU-RS025 | Une adresse facturation comprend : une rue, un numéro de rue, un code postal, une ville, un pays. |

### Etablissement

|  |  |
| --- | --- |
| BU-RS026 | Un établissement possède un nom, un type, une adresse. |
| BU-RS027 | Un établissement peut posséder une photo, un site web et une adresse mail |
| BU-RS028 | Un établissement peut posséder une liste d’horaires |
| BU-RS029 | Un établissement peut vendre des bières |

### Typeétablissement

|  |  |
| --- | --- |
| BU-RS030 | Il existe 4 types : bars, restaurant, magasin spécialisé, brasserie |

### Horaire

|  |  |
| --- | --- |
| BU-RS031 | Un horaire possède un jour de la semaine, une heure de début et une heure de fin |
| BU-RS032 | Un horaire est lié à un établissement |

### Utilisateur

|  |  |
| --- | --- |
| BU-RS033 | Un utilisateur possède une adresse mail et un mot de passe |

## Règles de validation

### Utilisateur

|  |  |
| --- | --- |
| BU-RV001 | L’adresse mail d’un utilisateur est unique |
| BU-RV002 | L’adresse mail est valide en tant qu’adresse mail |
| BU-RV003 | Le mot de passe encodé lors de l’enregistrement est identique à la confirmation du mot de passe. |

### Client

|  |  |
| --- | --- |
| BU-RV004 | L’adresse mail du client est unique |
| BU-RV005 | L’adresse mail du client est valide en tant qu’adresse mail |
| BU-RV006 | Le mot de passe encodé lors de l’enregistrement est identique à la confirmation du mot de passe. |
| BU-RV007 | Un client travaille pour une organisation |

### GroupAdmin

|  |  |
| --- | --- |
| BU-RV008 | Un GroupAdmin est automatiquement validé |

### Organisation

|  |  |
| --- | --- |
| BU-RV009 | L’adresse mail et le numéro de tva de l’organisation doivent être valides. |
| BU-RV010 | Le nom de l’organisation est unique |

### Bière

|  |  |
| --- | --- |
| BU-RV011 | Un utilisateur ne peut rajouter que 5 bières par jour |
| BU-RV012 | Le nom d’une bière est unique |
|  | Un client ne peut rajouter que 50 bières par jour |
|  | Pour désactiver une bière, celle-ci doit être validée |
|  | Pour supprimer une bière aucune facture ne peut reprendre la bière et la bière ne peut pas être validée. |

### Etablissement

|  |  |
| --- | --- |
| BU-RV013 | Pour supprimer un établissement aucune facture ne peut reprendre l’établissement ou encore être une brasserie possédant des bières |

### Brasserie

|  |  |
| --- | --- |
| BU-RV013 | Le nom d’une brasserie est unique |

### Horaire

|  |  |
| --- | --- |
| BU-RV014 | Il ne peut pas avoir pas avoir 2 horaires se chevauchant pour un même établissement |

### Avis

|  |  |
| --- | --- |
| BU-RV015 | Un avis doit avoir au moins 3 saveurs avec des notes différentes de zéro |
|  | Un avis dans l’Application Mobile doit avoir une note globale |
|  | Un avis dans l’Application Web ne peut pas avoir de note globale |
|  | Un avis, s’il en a une, doit avoir une note globale comprise entre 0 et 5 | Un avis, s’il en a une, doit avoir une note globale comprise entre 0 et 5 |
| BU-RV017 | Un utilisateur ne peut remettre que 2 avis pour une même bière sur la même journée. |
|  | Il ne peut y avoir qu’un seul avis actif par utilisateur et par bière |

### Vue des achats

|  |  |
| --- | --- |
| BU-RV018 | Il doit y avoir au moins un achat de vues sponsorisées |

### Adresse de facturation

|  |  |
| --- | --- |
| BU-RV018 | Tous les champs doivent être remplis |

### Plan tarifaire

|  |  |
| --- | --- |
| BU-RV018 | Tous les champs doivent être remplis |
|  | La date de début doit être antérieure à la date de fin |

### Rôles

|  |  |
| --- | --- |
| BU-RV018 | Seul un rôle sans utilisateur peut être supprimé |

## Règles de calcul

### Vues

|  |  |
| --- | --- |
| BU-RC001 | Chaque affichage d’une annonce pub décompte le nombre de vue payée |
| BU-RC002 | L’établissement sponsorisé affiché est choisi aléatoirement parmi les établissements ayant payé pour une vue et étant ouverts   * Si aucun établissement avec une vue payante est ouvert, un établissement fermé est alors affiché. |

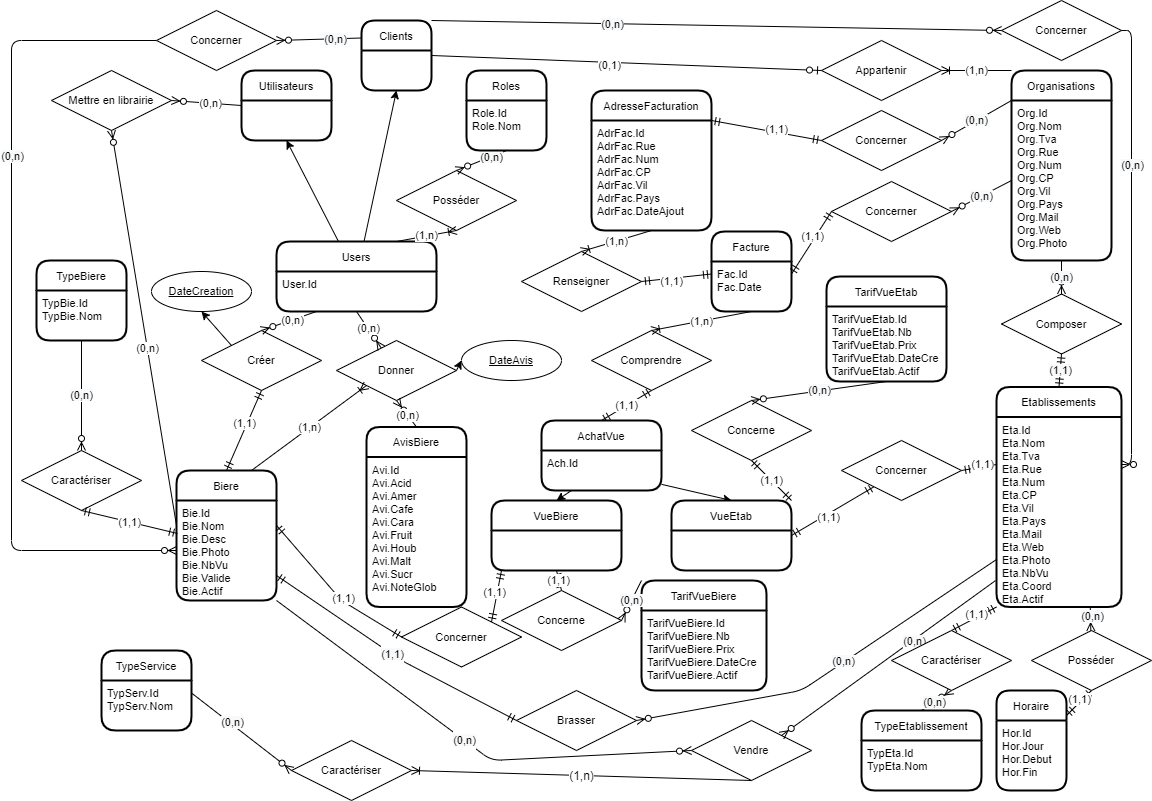
### Classement des bières

|  |  |
| --- | --- |
| BU-RC003 | Un mois est défini par 30 jours calendrier dans le classement des bières |

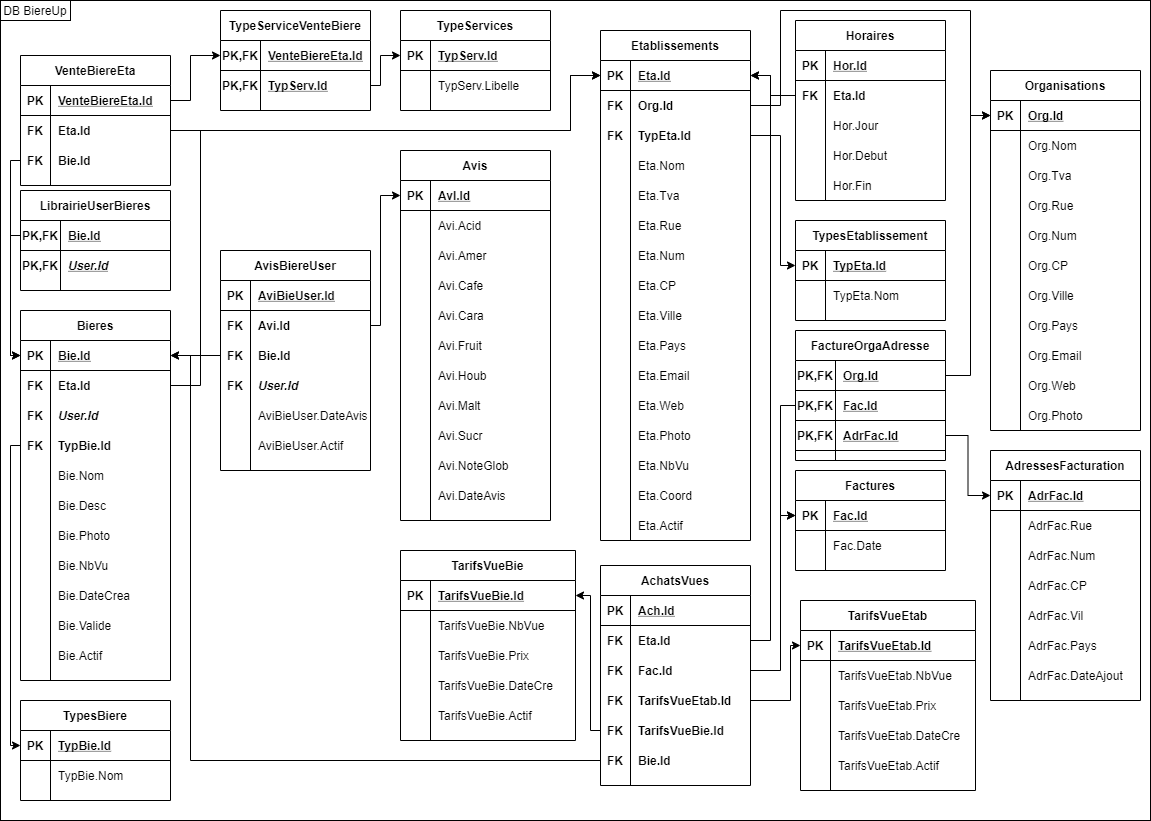
### Suggestions

|  |  |
| --- | --- |
| BU-RC004 | Une suggestion basée sur une bière a comme conditions :   * Bière du même type * & Bière n’ayant pas de note trop éloignée pour chaque goût (+1/–1) * & Bière présente dans un rayon de 20 km |
| BU-RC005 | Une suggestion basée sur les bières avec les meilleures notes du mois a comme condition :   * Bière qui est dans le top 10 des notes des utilisateurs dans le mois |
| BU-RC007 | Une suggestion basée sur les goûts de l’utilisateur et les meilleures du mois a comme condition :   * Bière dont le type est dans la bibliothèque de l’utilisateur * & Bière qui est dans le top 10 de ce type par les côtes des utilisateurs dans le mois * & Bière qui n’est pas dans la bibliothèque de l’utilisateur |
| BU-RC008 | Une suggestion surprise a comme condition :   * Bière aléatoire présente dans un rayon de 20km |

# Schéma Entité-Association



# Schéma relationnel



# Entités

## Bières

Une bière possède un nom et une brasserie.

Une bière possède un type et 8 saveurs notées sur une échelle de 0 à 5, 0 pour une bière qui ne contient pas la saveur et 5 pour une bière avec un goût très prononcé. Cette valeur est définie par les utilisateurs et/ou le client. Lorsque le client crée une bière, il peut noter les saveurs (voir plus loin).

Une bière peut posséder une note globale

Une bière peut posséder une description et une photo

# Analyse technique

## Technologies proposées

### Bases de données

Les 4 bases de données seront gérées via SQL Server. L’implémentation initiale est réalisée en Entity Framework Core avec des procédures stockées, elles, réalisées via SQL Server.

### API

Les différents accès aux bases de données seront réalisés via des Web API en C# qui utiliseront elles-mêmes des procédures stockées, à l’exception de la base de données ASP NET Identity qui sera en accès direct pour la composante IdentityServer

#### API Bieres

Point d’accès pour la base de données Bières

#### API Factures

Point d’accès pour la base de données Factures

#### API Organisations

Point d’accès pour la base de données Organisations

#### API VIsion

API de Google pour la reconnaissance des images

#### VATLayer

API utilisée pour la vérification de la validité du numéro de TVA encodé

#### API GeoCoding

L’API de de Google sera utilisée pour cela

### Site Web

Le site internet sera réalisé en MVC .Net Core 3.1 – Angular ? Razor ?

#### Validation

La validation sera réalisée via Fluent Validation

#### Création PDF

Nuget à creuser : trouver une solution gratuite ou passer via node.js ?

#### Paiements

Mullie à vérifier licence étudiant

#### Notifications à l’utilisateur

Les notifications à l’utilisateur seront réalisées via ToastR

### Application Mobile

L’application mobile sera développée à l’aide de Flutter en Dart

#### Carte des établissements

Google Map sera utilisé comme solution

### Authentification

L’authentification se fera IdentityServer ainsi que les modifications de rôles et d’utilisateurs

## Architecture applicative

