Annexes

I. Cahier des Charges

a. Users Stories

En tant que	J'ai besoin de pouvoir	Afin de
Visiteur	Créer un compte	Accéder aux fonctionnalités réservées aux adhérents
	Consulter le catalogue des recettes	Trouver une recette susceptible de m'intéresser
	Consulter une fiche recette	La cuisiner
	Consulter le catalogue de films & séries	Trouver des recettes en rapport avec une œuvre particulière
	Consulter la fiche d'un film ou série	Consulter les recettes associées
Adhérent - Non connecté	Me connecter	Accéder aux fonctionnalités réservées aux adhérents
Adhérent - Connecté	Créer une fiche recette	Contribuer au catalogue
	Lister mes recettes publiées	Gérer mes recettes
	Modifier une fiche recette	Mettre à jour ou corriger ma recette
	Supprimer une fiche recette	Garder le contrôle sur le contenu que je partage
	Modifier infos de mon compte	Sécuriser mon compte, exercer mes droits conformément au RGPD
	Supprimer mon compte	Garder le contrôle de mes données
Admin	Modifier compte adhérent	Sanctionner les abus (censure ou bannissement)
	Supprimer compte adhérent	Sanctionner les abus (censure ou bannissement)
	Modifier recette	Maintenir un contenu de qualité
	Supprimer recette	Maintenir un contenu de qualité

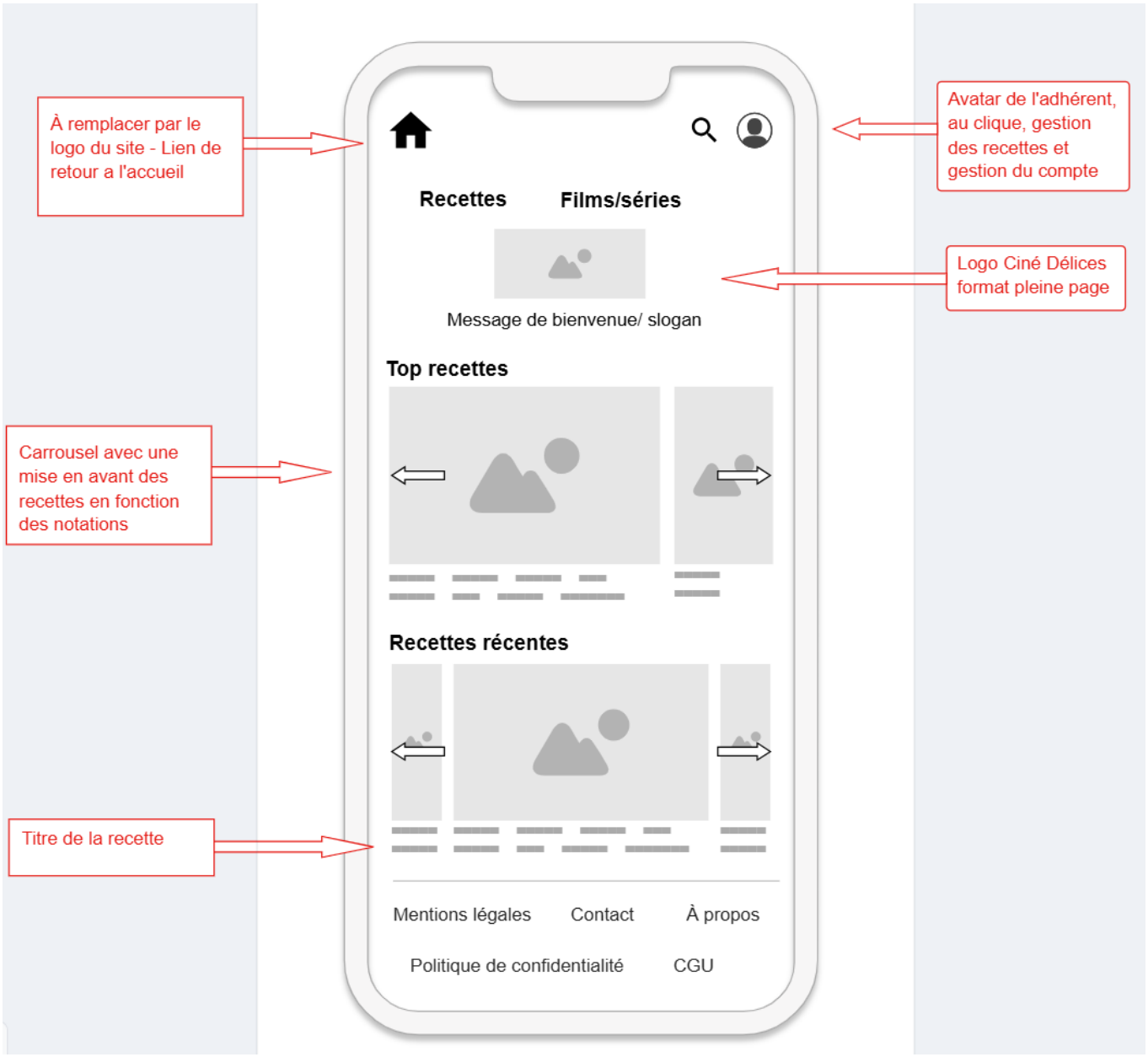
b. Routes Back-End

Entité	Méthode	Endpoint	Rôle
/	POST	/api/auth/login	Route d'authentification (midleware)
Utilisateurs	POST	/api/users/	Récupérer tous les utilisateurs
	GET	/api/users/id/:id	Récupérer un utilisateur par ID
	GET	/api/users/email/:email	Récupérer un utilisateur par email
	GET	/api/users/me	Récupérer l'utilisateur authentifié
	POST	/api/users/	Créer un utilisateur
	PATCH	/api/users/:id	Modifier un utilisateur
	DELETE	/api/users/:id	Supprimer un utilisateur
Recettes	GET	/api/recipes	Récupérer toutes les recettes
	GET	/api/recipes/:id	Récupérer une recette par ID
	GET	/api/my-recipes	Récupérer les recettes d'un utilisateur authentifié
	POST	/api/recipes	Créer une recette
	PATCH	/api/recipes/:id	Modifier une recette par ID
	DELETE	/api/recipes/:id	Supprimer une recette par ID
Films	GET	/api/movies	Récupérer la liste des films stocké en BDD
Categories	GET	/api/categories	Récupérer toutes les catégories
	GET	/api/categories/:id	Récupérer une catégorie par ID
	POST	/api/categories	Créer une catégorie

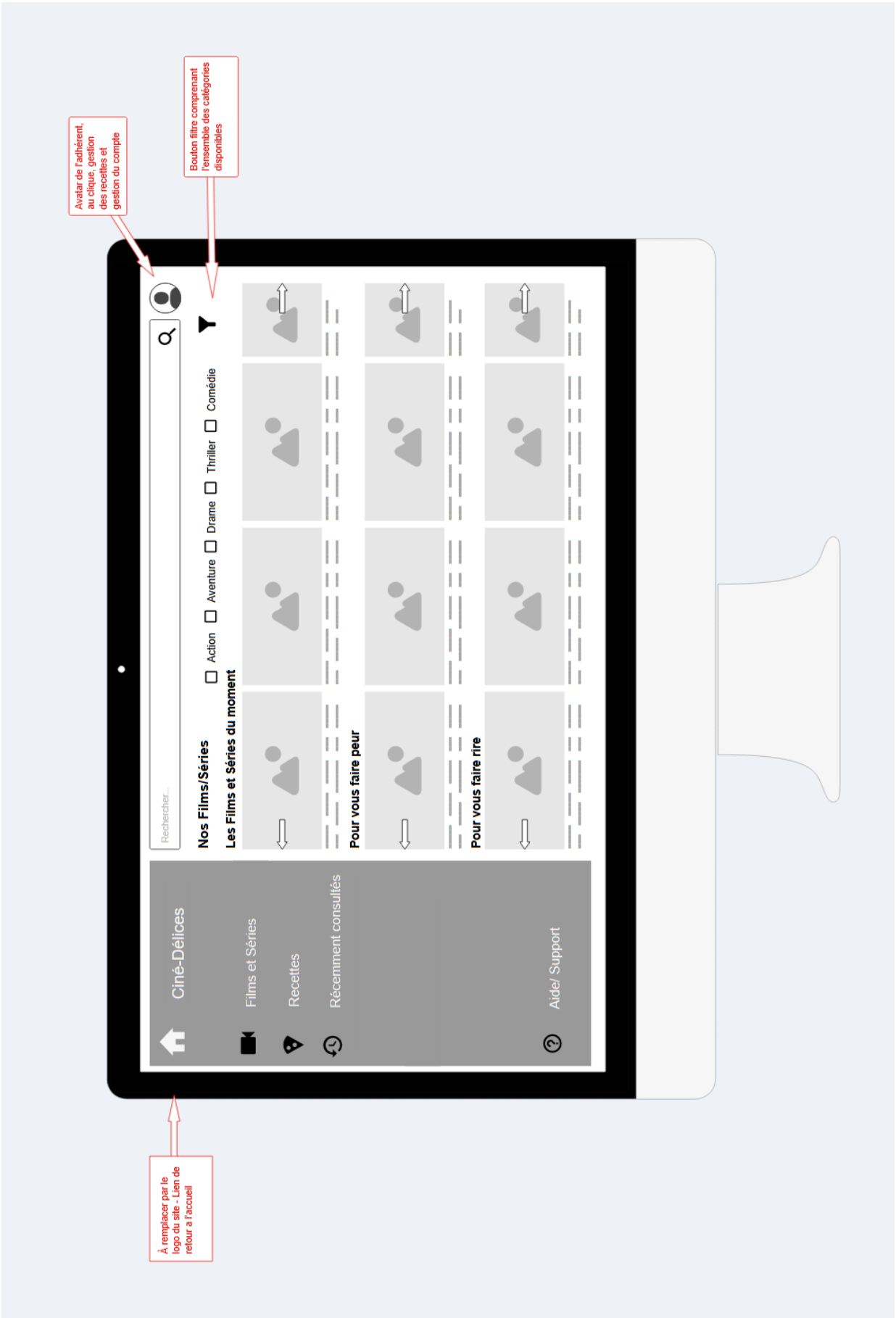
Entité	Méthode	Endpoint	Rôle
Categories	PATCH	/api/categories/:id	Modifier une catégorie par ID
	DELETE	/api/categories/:id	Supprimer une catégorie par ID
Etapes	GET	/api/steps	Récupérer toutes les étapes
	GET	/api/steps/:id	Récupérer une étape par ID
	POST	/api/steps	Créer une étape
	PATCH	/api/steps/:id	Modifier une étape par ID
	DELETE	/api/steps/:id	Supprimer une étape par ID
Genres	GET	/api/genres	Récupérer tous les genres
	GET	/api/genres/:id	Récupérer un genre par ID
	POST	/api/genres	Créer un genre
	PATCH	/api/genres/:id	Modifier un genre par ID
	DELETE	/api/genres/:id	Supprimer un genre par ID

II. Exemples Wireframe

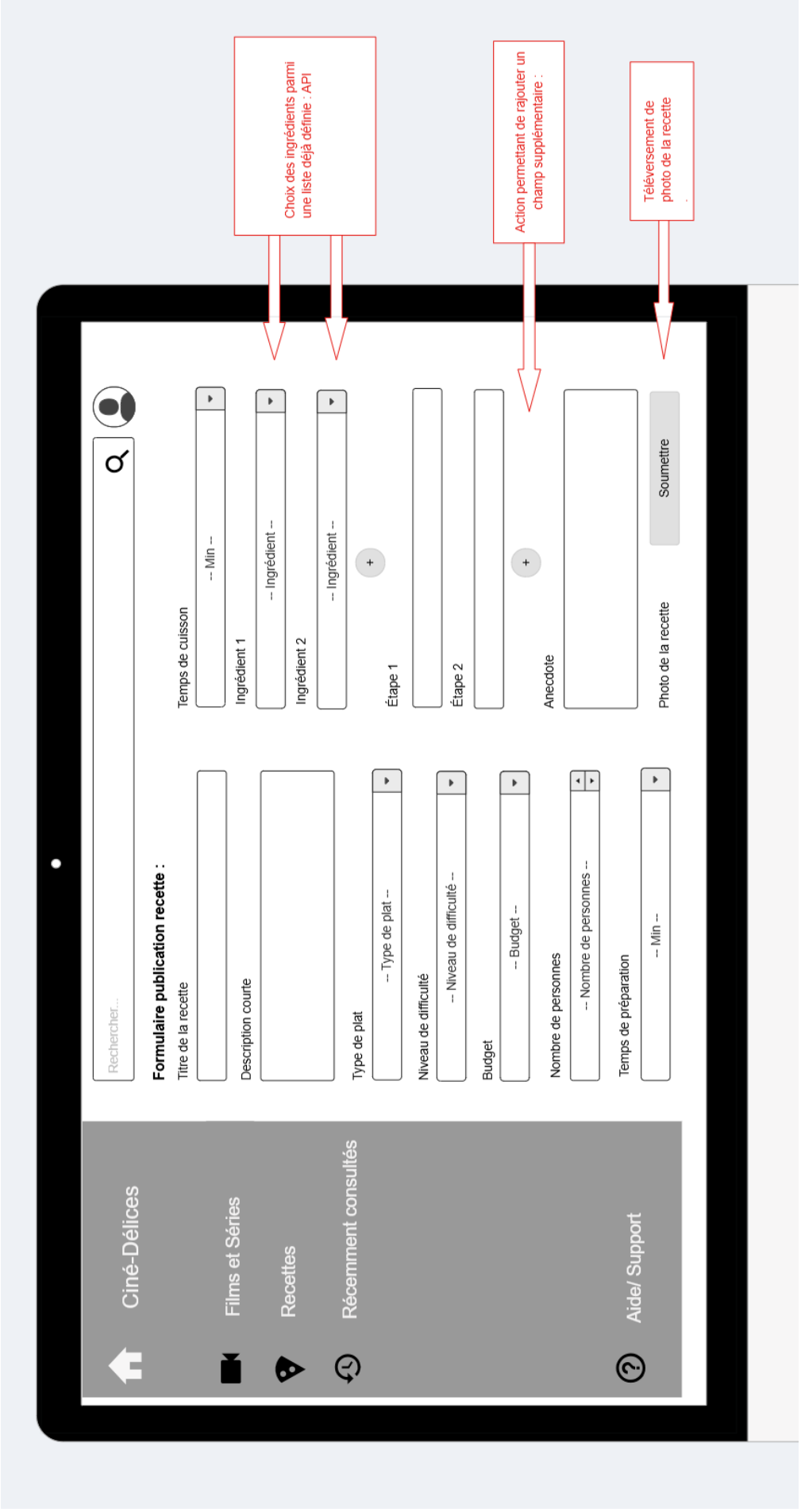
1. Accueil version Mobile



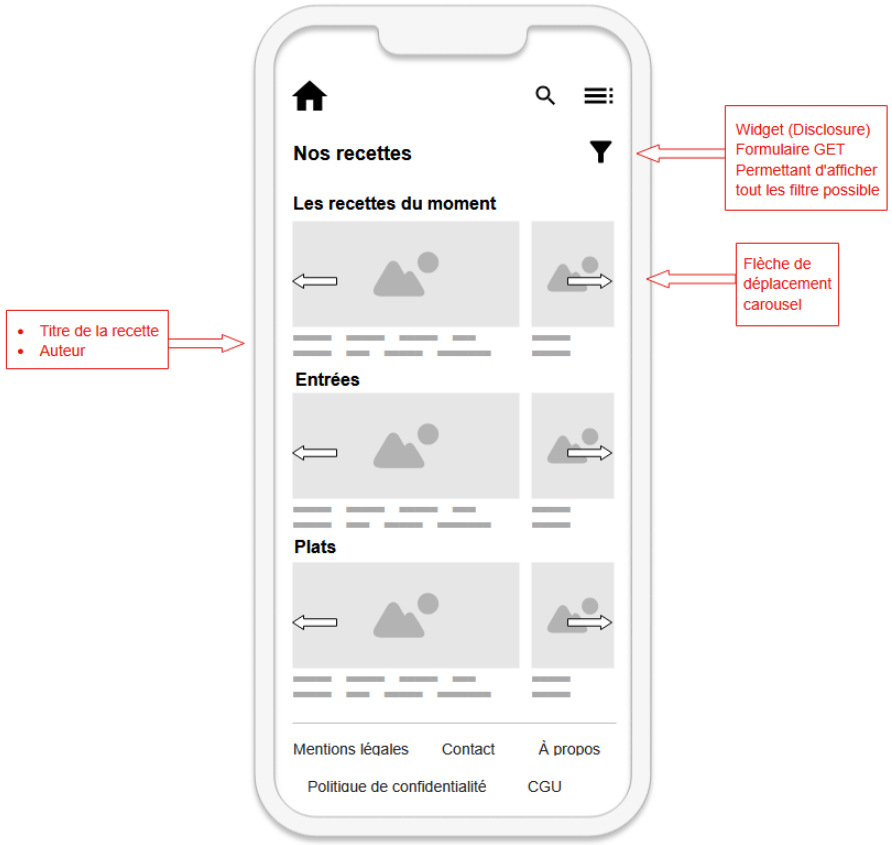
2. Catalogue de Films/Séries version Desktop 3.



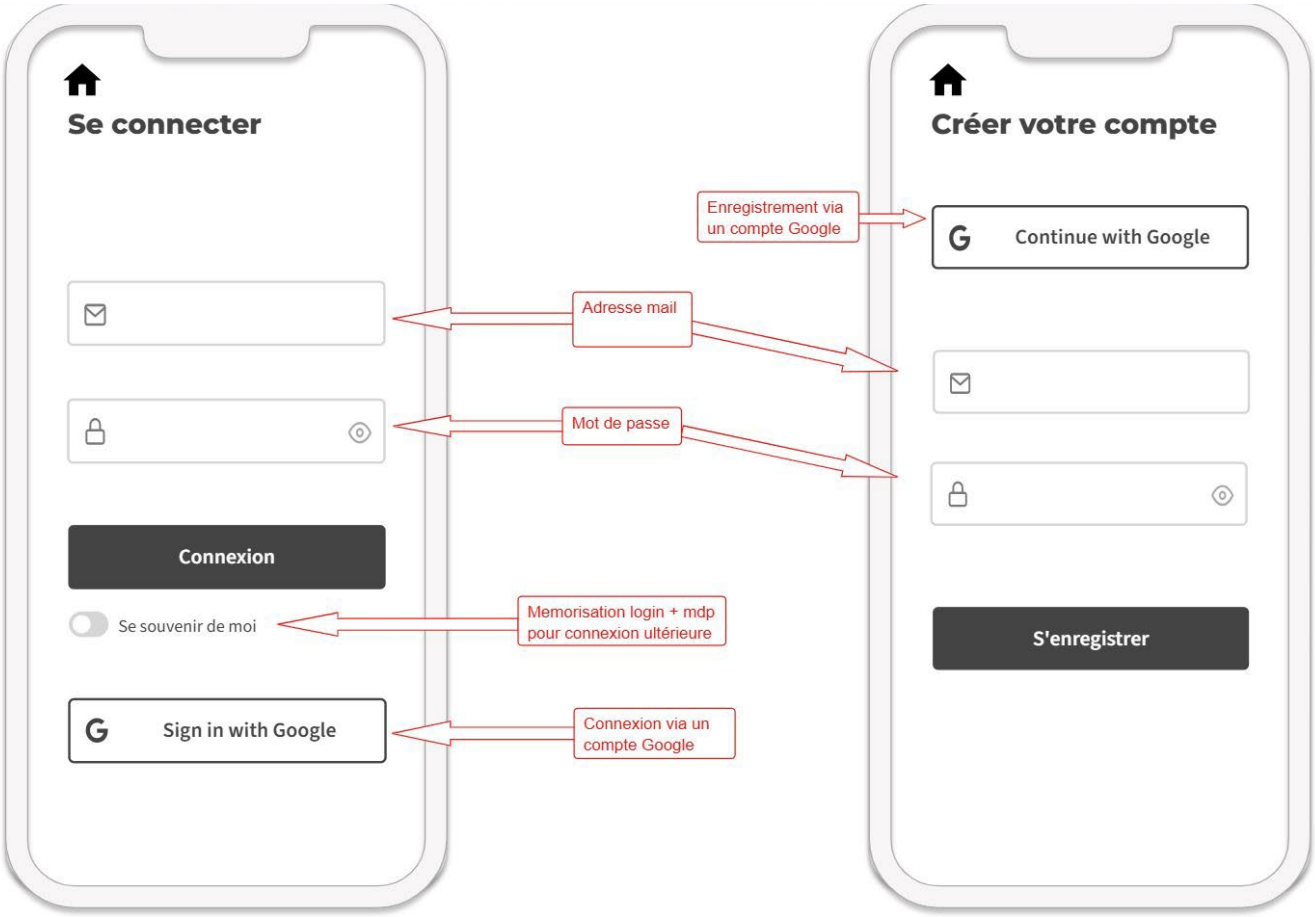
4. Formulaire de recette version Desktop



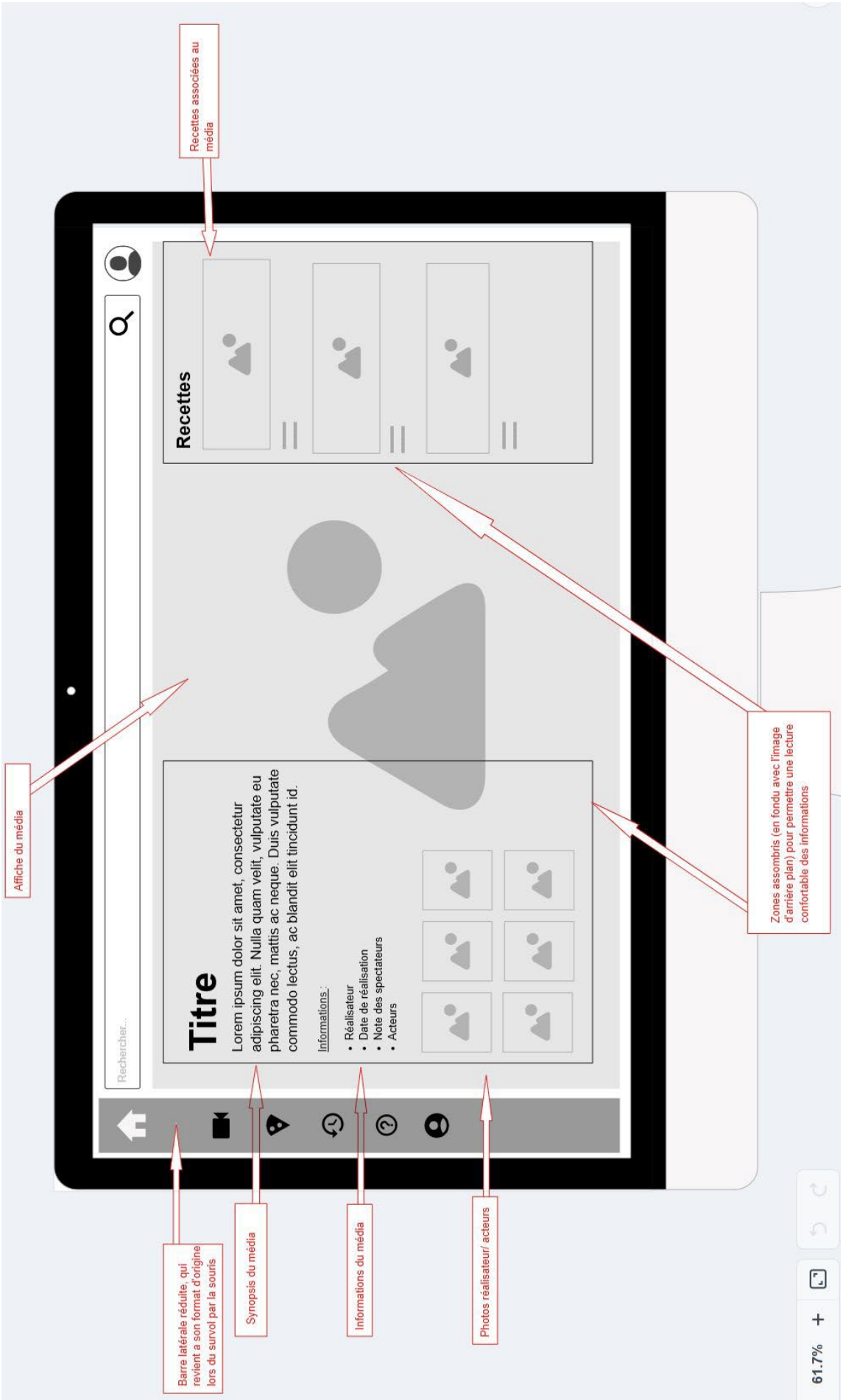
5. Catalogue de recette de cuisine version Mobile :



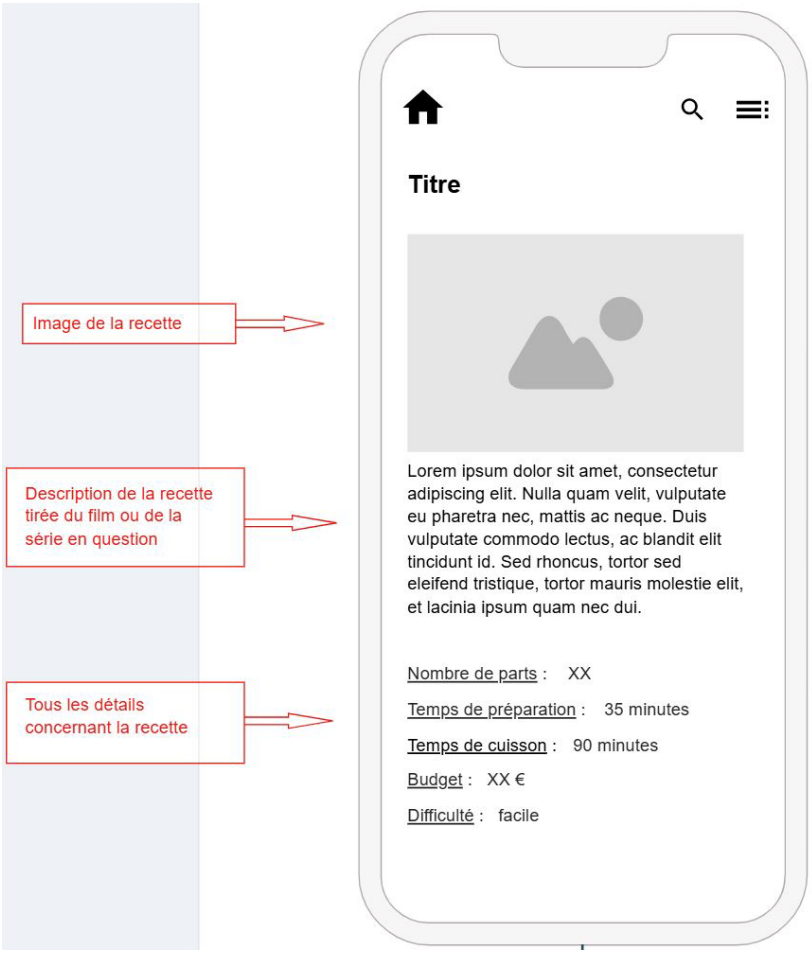
6. Formulaire de connexion et de création de compte version Mobile :



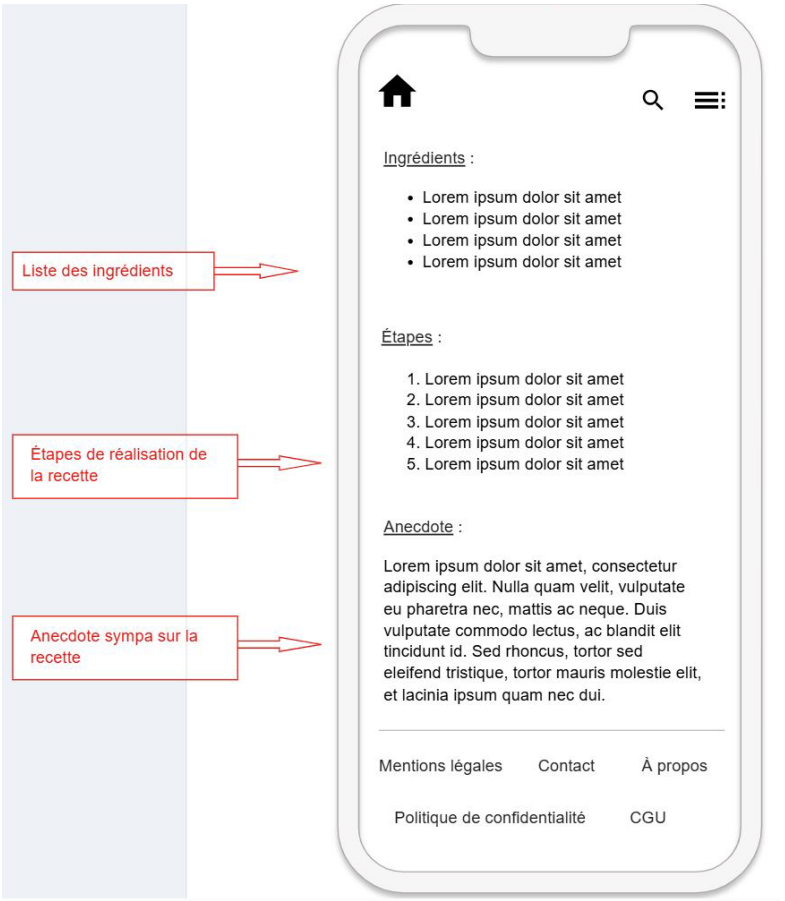
7. Page détails d'un film/série version Desktop :



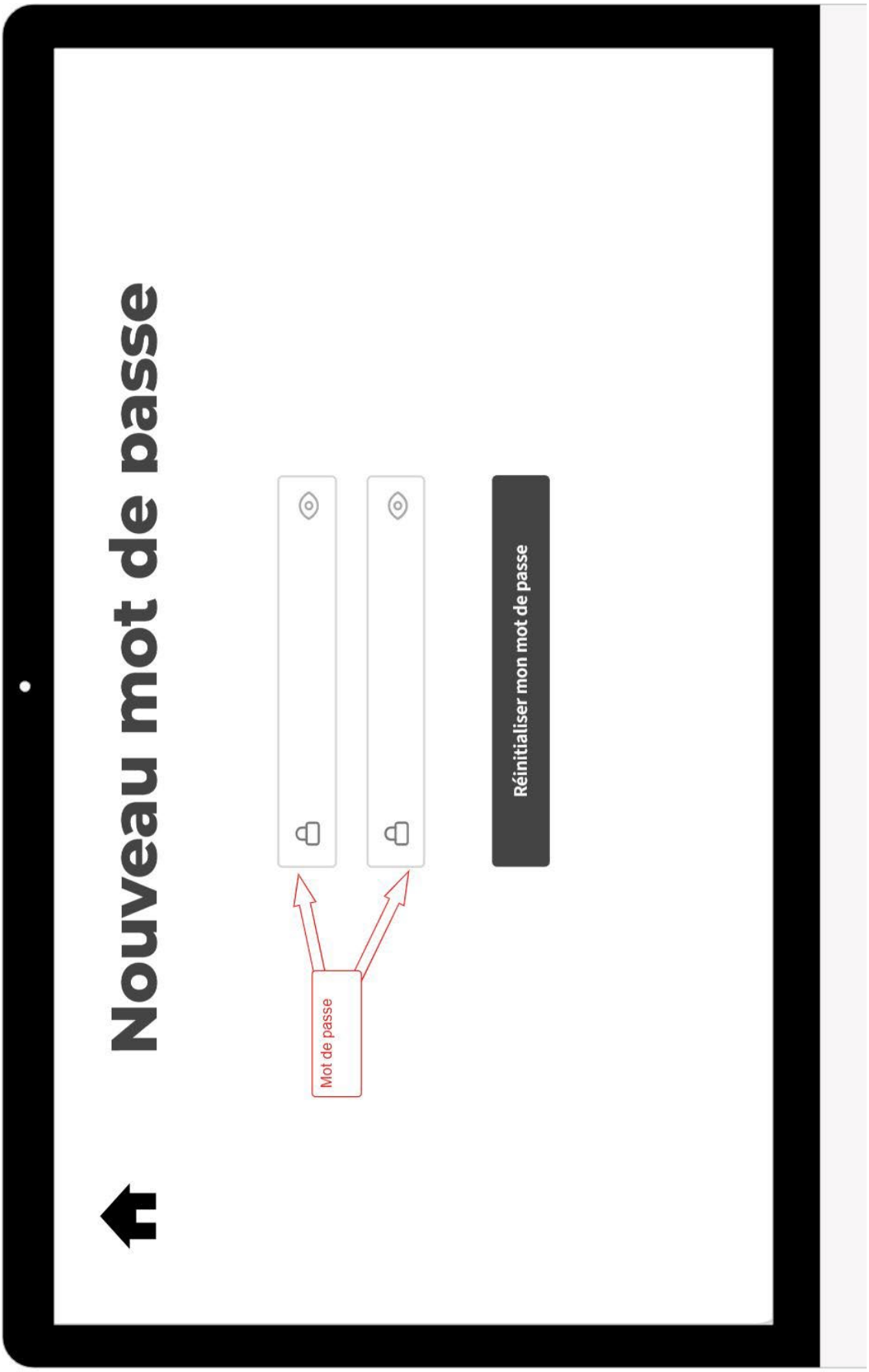
8. Haut de page fiche détails d'une recette version Mobile :



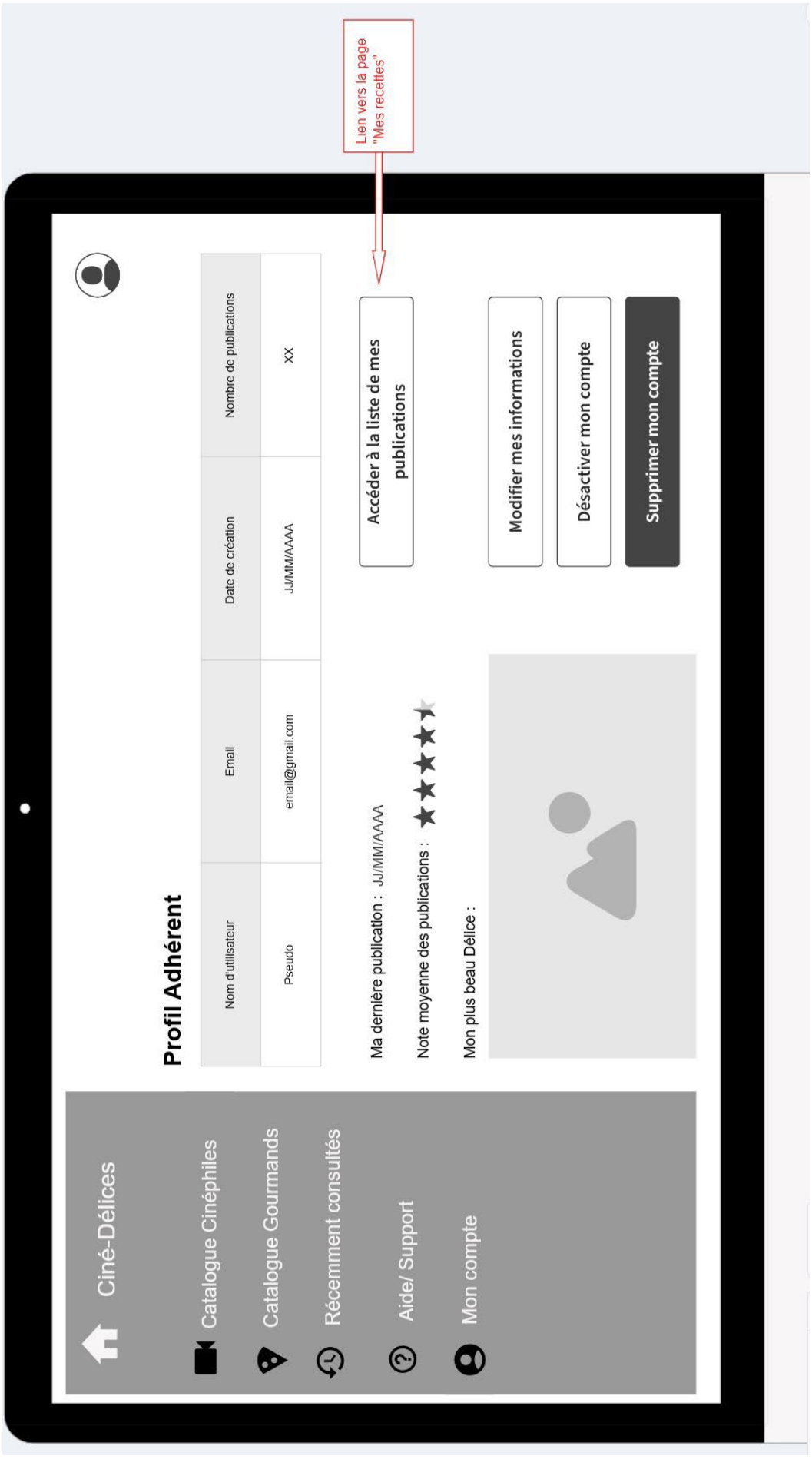
9. Bas de page fiche détails d'une recette version Mobile :



10. Formulaire de demande de nouveau mot de passe version Desktop :



11. Page de gestion de compte utilisateur version Desktop :



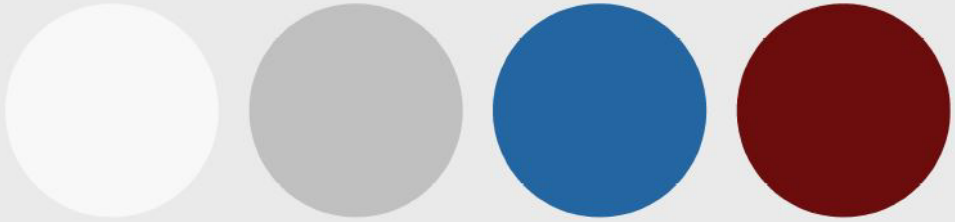
Charte Graphique

Couleurs :

Primary
#3CA0C8

Background-Color
#07202A
#18576E

Secondary
#2A2A2A



Color-Text
#F8F8F8

Secundary-
Text-Color
#C0C0C0

Primary-
Button-Color
#2366A1

Secondary-
Button-Color
#6B0D0D

Typographie :

Raleway

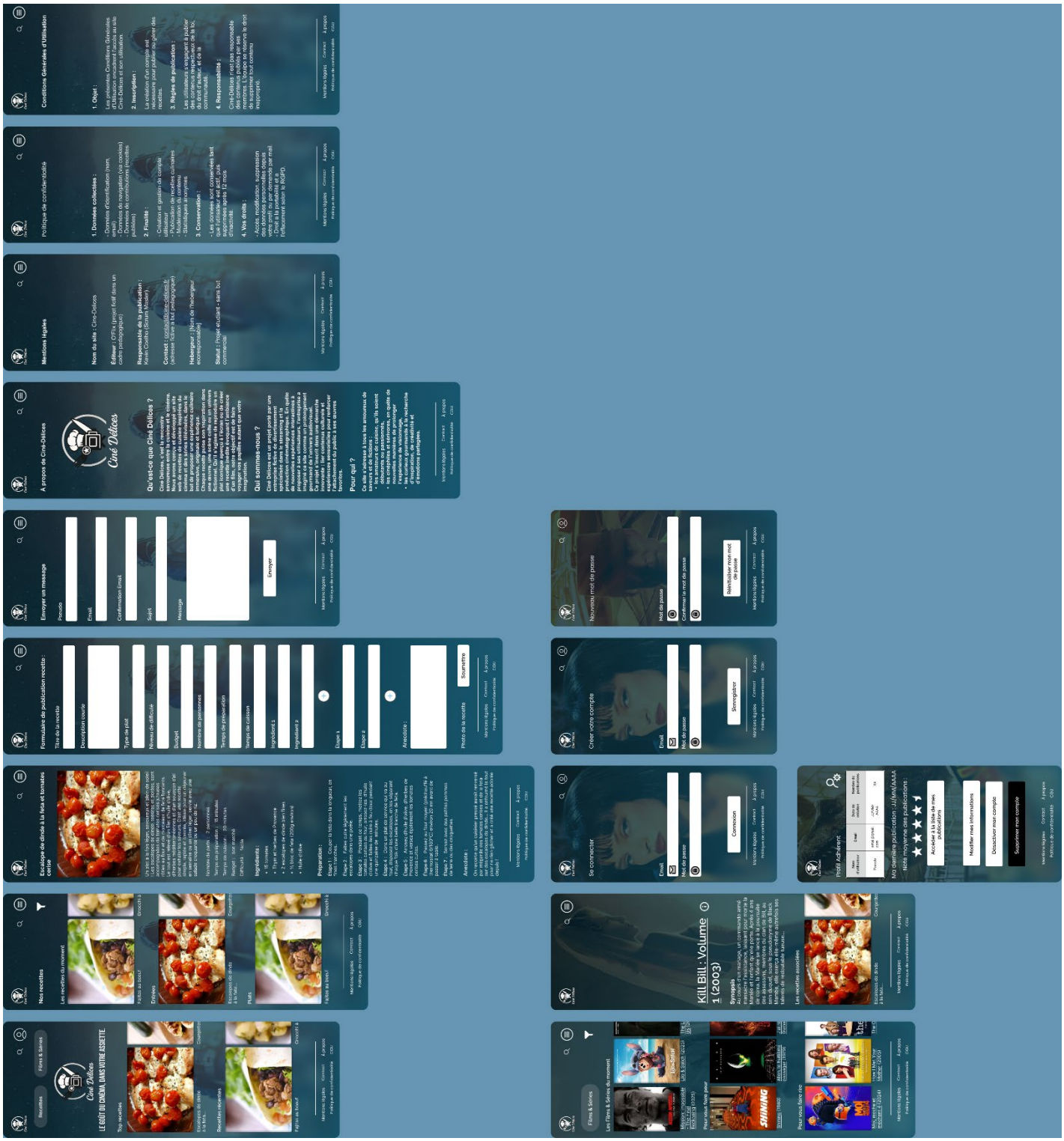
Regular 500	**Bold 700**
Viatorum inveniretur intersaepientes opibus igitur.	**Viatorum inveniretur intersaepientes opibus igitur.**

Inter

Regular 500	**Bold 700**
Viatorum inveniretur intersaepientes opibus igitur.	**Viatorum inveniretur intersaepientes opibus igitur.**

IV. Maquette graphiqu

1. Vue d'ensemble :



V. Exemple Maquette Graphique :

1. *Accueil :*

2. *Catalogue de recette :*

3. *Catalogue de films et séries :*

4. *Fiche détails d'un film/série :*

VI. Extrait de code

a. Front

Composant du formulaire de connexion au format JSX :

```jsx
export default function SignupForm() {
    return (
        <form
            className="account__form"
            method="POST"
        >
            <fieldset>
                {/* Ajout d'une légende, qui sera "caché" visuellement
                mais accessible via une lisseuse d'écran*/}
                <h1>Créer votre compte</h1>
                <legend className="sr-only">
                    Formulaire de création de compte :
                </legend>
                <div className="account__form__group">
                    <label htmlFor="username" className=`account__form__label >
                        Nom d'utilisateur
                    </label>
                    <input
                        className="account__form__input"
                        type="text"
                        id="username"
                        name="username"
                        required
                    />
                </div>
                <div className="account__form__group">
                    <label htmlFor="email" className=`account__form__label>
                        <span>Email</span>
                        {/* Indication sur le format attendu accessible */}
                        <span className="sr-only">Exemple : nom@domaine.extension</span>
                    </label>
                    <input
                        className="account__form__input"
                        type="email"
                        id="email"
                        name="email"
                        required
                    />
                </div>
                <div className="account__form__group">
                    <label htmlFor="password" className=`account__form__label>
                        <span>Mot de passe</span>
                        <span className="sr-only">Veuillez insérer votre mot de passe :</span>
                        <span id="passwordHelp">Doit contenir au moins 12 caractères, une
majuscule, un chiffre et un caractère spécial.</span>
                    </label>
                    <input
                        className="account__form__input"
                        type="password"
                        id="password"
                        name="password"
                        aria-describedby="passwordHelp"
                        minLength={12}
                        required
                    />
                </div>
                <div className="account__form__group">
                    <label htmlFor="retype-password" className=`account__form__label`>
                        <span>Confirmation de votre mot de passe</span>
                        <span className="sr-only">Veuillez valider votre mot de passe :</span>
                    </label>
                    <input
                        className="account__form__input"
                        type="password"
                        id="retype-password"
                        name="retype-password"
                        required
                    />
                </div>
                <button type="submit" className="account__form__button">
                </button>
            </fieldset>
        </form>
    );
}
```

AccountForm.scss :

```scss
@use "../../vars";

.main__form {
    display: flex;
    justify-content: center;

    .account__form__wrapper {
        background-color: rgba(0, 0, 0, 0.1);
        width: 80%;
        padding: 1.5rem;
        border-radius: 10px;

        h1 {
            text-align: center;
            margin-bottom: 2rem;
        }

        /**
        * Messages importants pour la bonne complétion du formulaire
        * Comme l'obligation de remplir certains champs
        */
        .account__form__info {
            margin: 0 0 1.5rem 1rem;
            font-size: .8rem;
            font-style: italic;
            color: vars.$text-light;
        }

        // Message en cas d'erreur
        .error__indication {
            margin: 1rem 0;
            color: #ff0000;
            font-size: 1.2rem;
            font-style: italic;
            text-align: center;
        }

        // Conteneur label/input
        .account__form__group {
            position: relative;
            margin: 1rem 0;

            // Positionnement du label
            .account__form__label {
                position: absolute;
                top: 38%;
                left: 1rem;
                color: vars.$text-secondary;
                transition: all .4s ease-in-out;
            }

            .account__form__input {
                box-sizing: border-box;
                border: 2px solid vars.$border-color-input;
                border-radius: 30px;
                width: 100%;
                margin: 1rem 0;
                padding: 1.4rem;
                color: vars.$text-light;
                background-color: rgba(255, 255, 255, .1);
            }

            // Classe permettant la transition du label en dehors de l'input
            .is-active {
                top: 0;
                color: vars.$text-light;
                transform: translateY(-65%) translateX(5%);
            }
        }

        // Boutton de formulaire
        .account__form__button {
            position: relative;
            margin-top: 4rem;
            padding: 1.4rem 0;
            border: 2px solid vars.$border-color-input;
            border-radius: 40px;
            background-color: vars.$button-primary-color;
            transition: .3s;

            &:hover,
            &:focus
            {
                background-color: vars.$button-secondary-color;
                outline: none;
            }
        }
    }
}
```

Extrait du fichier `home.scss` :

```scss
.hero {
    display: flex;
    flex-direction: column;
    text-align: center;
    align-items: center;

    .logo_fullscreen {
        width: 80%;
        max-width: 300px;
    }

    .catchphrase {
        font-size: 1.5rem;
        color: vars.$white;
        font-family: vars.$font-main;
        margin-bottom: 3rem;
    }

    // Media Queries
    // Breakpoint : largeur d'au moins 768px
    @include medium {
        flex-direction: column;

        .logo_fullscreen {
            max-width: 400px;
        }

        .catchphrase {
            font-size: 1.7rem;
        }
    }
    // Breakpoint : largeur d'au moins 1024px
    @include large {
        flex-direction: column;

        .logo_fullscreen {
            max-width: 450px;
        }

        .catchphrase {
            font-size: 2rem;
        }
    }
}
```

Hook personnalisé du formulaire de connexion au format JSX :

```jsx
import { useContext, useState } from "react"
import { useNavigate } from "react-router-dom";

import { AuthContext } from "../Authentication";

export default function useLogin() {
    // Définition des variables d'état pour chaque champs, etat initial = chaîne de caractère vide
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    // Variable d'état qui gérera le message d'erreur à afficher
    const [message, setMessage] = useState("");
    /**
     * On fait appel à notre fonction useNavigate pour permettre la redirection
     * @link https://api.reactrouter.com/v7/functions/react_router.useNavigate.html
     */
    const navigate = useNavigate()

    // Utilisation de la fonction
    const { login } = useContext(AuthContext)
    // Handler pour la soumission du formulaire
    const handleSubmit = async (e) => {
        e.preventDefault();
        // conditions à la soumission pour être connecté
        try {
            const payload = {
                email,
                password
            }
            const response = await fetch('http://localhost:3000/api/auth/login', {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                    "charset": "utf-8"
                },
                body: JSON.stringify(payload)
            })
            // si les conditions ne sont pas remplies
            if (!response.ok) {
                const error = await response.json();
                throw new Error(error.message)
                // utilisation de useNavigate dans le router pour la redirection
            } else {
                // Récupération du token, qui se trouve dans le corps de notre requêtes
                const data = await response.json()
                const token = data.token
                // On le stock dans le localStorage
                login(token)
                alert("Connexion réussie !");
                navigate('/my-account')
            }
            setMessage('')
        } catch (error) {
            setMessage(error.message)
        }
    }

    return {
        email,
        setEmail,
        password,
        setPassword,
        message,
        handleSubmit
    }
}
```

Composant du formulaire de connexion au format JSX :

```jsx
import { useState } from "react";
import useLogin from "../../../Hook/useLogin";

export default function LoginForm() {
    // On importe nos variables d'état ainsi que nos handler de notre hook personnalisé
    const { email, setEmail, password, setPassword, message, handleSubmit } = useLogin();
        /**
         * Variable d'état qui permet de détecter si le focus sur le champs est effectué
         * Par défaut, valeur à false
         */
        const [focusState, setFocusState] = useState({
            email: false,
            password: false,
        })
        /**
         * Handler permettant la gestion des variable d'état au focus de nos champs
         * @param {string} field - Chaîne de caractère représentant la clé de notre objet stocké dans notre variable d'état
         */
        const handleFocus = (field) => {
            setFocusState((prev) => ({
                ...prev, [field]: true
            }))
        }
        // Handler lors de la perte de focus de nos champs
        const handleBlur = (field) => {
            setFocusState((prev) => ({
                ...prev, [field]: false
            }))
        }
        /**
         * On affecte la valeur correspondante (true ou false)
         * En fonction du résultat du handler
         */
        const isEmailActive = focusState.email;
        const isPasswordActive = focusState.password;

    return (
        <form
            className="account__form"
            method="POST"
            onSubmit={handleSubmit}
        >
            {/* Si la valeur de "message" est "true", alors on l'affiche */}
            {message && <p className="error__indication">{message}</p>}
            {/* Ajout d'un fieldset, pour mieux structurer sémantiquement le formulaire */}
            <fieldset>
                {/* Ajout d'une légende, qui sera "caché" visuellement, mais accessible via une lisseuse d'écran*/}
                <legend className="sr-only">Formulaire de connexion :</legend>
                <span className="account__form__info">Tout les champs sont obligatoire</span>
                <div className="account__form__group">
                    <label htmlFor="email" className={`account__form__label ${isEmailActive || email ? "is-active" : ""}`}>
                        Email
                        {/* Indication sur le format attendu accessible */}
                        <span className="sr-only">Exemple : nom@domaine.extension</span>
                    </label>
                    <input
                    <input
                        className="account__form__input"
                        type="email"
                        id="email"
                        name="email"
                        // prise en compte de la valeur du champ une fois rempli, ici spécifique à l'email
                        // puis par la suite adapté en fonction de la valeur des autres champs
                        onChange={(e) => setEmail(e.target.value)}
                        onFocus={() => handleFocus("email")}
                        onBlur={() => handleBlur("email")}
                        value={email}
                        required
                    />
                </div>
                <div className="account__form__group">
                    <label htmlFor="password" className={`account__form__label ${isPasswordActive || password ? "is-active" : ""}`}>
                        Mot de passe
                    </label>
                    <input
                        className="account__form__input"
                        type="password"
                        id="password"
                        name="password"
                        onChange={(e) => setPassword(e.target.value)}
                        onFocus={() => handleFocus("password")}
                        onBlur={() => handleBlur("password")}
                        value={password}
                        minLength={12}
                        required
                    />
                </div>
                <button type="submit" className="account__form__button">
                    Connexion
                </button>
            </fieldset>
        </form>
    );
}
```

Contexte d'environnement - Authentification :

```jsx
/**
 * Gestion des routes autorisées avec React
 * Nous utiliserons ici les mécaniques de ``context``
 * @link https://fr.react.dev/reference/react/createContext
 * On va pouvoir créer un "environnement"
 * Qui nous permettra de véhiculer des données récurrentes
 * Sans avoir besoin de passer par des props
 */
import { createContext, useState, useEffect, useContext } from "react";
/**
 * On initialise un nouveau contexte qu'on exportera
 * @link https://fr.react.dev/reference/react/createContext#importing-and-exporting-context-from-a-file
*/
export const AuthContext = createContext(null);
// On crée un hook personnalisé pour accéder au contexte d'authentification
// Ce hook permet d'utiliser le contexte d'authentification dans n'importe quel composant
export function useAuth() {
    return useContext(AuthContext);
}
/**
 * Nous initialiserons une fonction qui contiendra le ``Provider``
 * Qui fournira les valeurs récupérées à tous les composants lisant ce contexte
 * @link https://fr.react.dev/reference/react/createContext#provider
 */
export function AuthProvider({ children }) {
    const [token, setToken] = useState(null);
    const [isAuthenticated, setIsAuthenticated] = useState(false);
    const [isLoadingUser, setIsLoadingUser] = useState(true);
    const [username, setUsername] = useState(null);
    const [userData, setUserData] = useState(null);

    // Fonction lancée après chargement du rendu du composant
    useEffect(() => {
        // On récupère le token stocké dans localStorage
        const storedToken = localStorage.getItem("token");
        // Si le token a été généré et récupéré
        if (storedToken) {
            // On met à jour nos variables d'état
            setToken(storedToken);
            setIsAuthenticated(true);
            // Appel à l'API pour récupérer le pseudo
            fetch("http://localhost:3000/api/users/me", {
                headers: { Authorization: `Bearer ${storedToken}` }
            })
                // On récupère les données de l'utilisateur
                .then(res => {
                    if (!res.ok) throw new Error("Erreur récupération user");
                    return res.json();
                })
                .then(data => {
                    // On met à jour le pseudo de l'utilisateur
                    setUsername(data.username);
                    // On stocke les données utilisateur dans l'état
                    setUserData(data);
                })
                .catch(err => console.error("Erreur init AuthContext :", err.message))
                //quoi qu'il arrive, on arrête l'état de chargement (isLoadingUser) après la tentative de récupération du pseudo.
                .finally(() => setIsLoadingUser(false));
            // Si le token n'est pas présent, on met à jour l'état de chargement
        } else {
            setIsLoadingUser(false);
        }
    }, []);
```

Contexte d'environnement - Authentification (suite) :

```
    /**
     * On souhaite lancer cette opération qu'une fois après le chargement du rendu
     * @link https://fr.react.dev/reference/react/useEffect#examples-dependencies
     */
    /**
     * Mise à jour des variables d'états à la connexion
     * @param {string} newToken - Token d'authentification
     */
    const login = async (newToken) => {
        setToken(newToken);
        setIsAuthenticated(true);
        localStorage.setItem("token", newToken);
        try {
            const response = await fetch("http://localhost:3000/api/users/me", {
                headers: { Authorization: `Bearer ${newToken}` }
            });

            if (!response.ok) throw new Error("Erreur récupération utilisateur");
            const data = await response.json();

            if (!data.is_active) {
                alert("Votre compte est désactivé. Veuillez contacter un administrateur.");
                localStorage.removeItem("token");
                setToken(null);
                setUserData(null);
                setIsAuthenticated(false);
                // Empêche la suite de la connexion
                return;
            }
            // Pour une mise à jour immédiate, sinon on devrait rafraichir la page pour avoir l'affichage du pseudo et le bouton de
          déconnexion
            setUsername(data.username);
            // On stocke les données utilisateur
            setUserData(data);
        } catch (error) {
            console.error("Erreur lors du login :", error.message);
            setIsLoadingUser(false);
        }
    };

    const logout = () => {
        localStorage.removeItem("token");
        setToken(null);
        setIsAuthenticated(false);
        setUsername(null);
        setUserData(null);
    };
    /**
     * Le contexte fonctionne comme un composant React
     */
    return (
        <AuthContext.Provider value={{ token, isAuthenticated, isLoadingUser, username, userData, login, logout }}>
            {children}
        </AuthContext.Provider>
    );
}
```