

Scénarios de Test RealWorld

Scénario 1 : Inscription et première connexion d'un utilisateur

Objectif

Vérifier qu'un nouvel utilisateur peut s'inscrire et se connecter avec succès

Prérequis

- Application RealWorld accessible
- Base de données vide ou avec données de test

Étapes

1. Accéder à la page d'inscription

- URL : `/#/register`
- Vérifier que le formulaire d'inscription s'affiche

2. Remplir le formulaire d'inscription

- Username : "admin"
- Email : "admin@example.com"
- Password : "SecurePass123"

3. Soumettre l'inscription

- Cliquer sur "Sign up"
- API attendue : `POST /api/users`

4. Vérifier la réponse d'inscription

- Code retour : 200
- Réponse contient :

```
{ "user": { "email": "test@example.com",  
            "token": "jwt.token",  
            "username": admin,  
            "bio": null,  
            "image": null }  
}
```

- Token JWT présent et valide

5. Vérifier la redirection

- L'utilisateur est redirigé vers la page d'accueil (`/#/`)
- Header contient le nom d'utilisateur et lien de déconnexion

Résultats attendus

- ☒ Utilisateur créé en base de données
- ☒ Token JWT généré et stocké (localStorage)
- ☒ Interface utilisateur connectée affichée

{

Données de test

Json

```
{
  "user": {
    "Username" : "admin",
    "Email" : "admin@example.com",
    "Password" : "SecurePass123!"
  }
}
```

Scénario 2 : Création et publication d'un article

Objectif

Vérifier qu'un utilisateur connecté peut créer et publier un nouvel article

Prérequis

- Utilisateur authentifié (token JWT valide)
- Accès à la page d'édition

Étapes

1. Naviguer vers l'éditeur d'article

- Cliquer sur "New Post" depuis la navigation
- URL : `/#/editor`

2. Remplir le formulaire d'article

- Title : "Mon Premier Article de Test"
- Description : "Ceci est un article de test pour valider la fonctionnalité"
- Body : "Contenu détaillé de l'article avec du markdown supporté"
- Tags : "test, automation, realworld"

3. Publier l'article

- Cliquer sur "Publish Article"
- API attendue : `POST /api/articles`

4. Vérifier la création

- Code retour : 200
- Article créé avec slug généré automatiquement
- Redirection vers la page de l'article

5. Valider l'affichage de l'article

- URL : `/#/article/{slug}`
- Titre, description, contenu correctement affichés
- Tags présents et cliquables
- Informations auteur (username, image)
- Boutons "Edit" et "Delete" visibles (propriétaire)

Résultats attendus

- ☒ Article persisté en base de données
- ☒ Slug généré à partir du titre
- ☒ Markdown rendu correctement
- ☒ Tags associés à l'article

Données de test

Json

```
{
  "article": {
    "title" : "Mon Premier Article de Test",
    "description" : "Ceci est un article de test pour valider la fonctionnalité",
    "body" : "Contenu détaillé de l'article avec du markdown supporté!",
    "tagList" : ["test", "automation", "realwolrd"]
  }
}
```

Scénario 3 : Parcours complet de gestion des favoris

Objectif

Vérifier le système de favoris (ajouter/retirer) et son impact sur le compteur





Prérequis

- 2 utilisateurs : auteur_article et lecteur_favori
- 1 article existant créé par auteur_article

Étapes

1. Se connecter en tant que lecteur_favori
 - Login avec credentials du lecteur
2. Consulter l'article
 - Naviguer vers l'article existant
 - Vérifier que `favorited: false` et `favoritesCount: 0`
3. Ajouter aux favoris
 - Cliquer sur le bouton "Favorite" (cœur)
 - API attendue : `POST /api/articles/{slug}/favorite`
4. Vérifier l'ajout aux favoris
 - Code retour : 200
 - `favorited: true` et `favoritesCount: 1`
 - Bouton visuel mis à jour (cœur rempli)
5. Retirer des favoris
 - Cliquer de nouveau sur le bouton "Unfavorite"
 - API attendue : `DELETE /api/articles/{slug}/favorite`
6. Vérifier la suppression
 - Code retour : 200
 - `favorited: false` et `favoritesCount: 0`
 - Bouton visuel remis à l'état initial
7. Vérifier la persistance
 - Rafraîchir la page
 - État des favoris maintenu correctement

Résultats attendus

-  Toggle favoris fonctionne correctement
-  Compteur se met à jour en temps réel
-  État persiste après rechargement
-  Seuls les utilisateurs authentifiés peuvent favoriser

Scénario 4 : Système de commentaires

Objectif

Tester l'ajout, affichage et suppression de commentaires sur un article

Prérequis

- Utilisateur authentifié
- Article existant avec URL accessible

Étapes

1. Accéder à un article

- Naviguer vers `/#/article/{slug}`
- Vérifier la section commentaires en bas de page

2. Ajouter un commentaire

- Saisir dans le champ texte : "Super article ! Très instructif."
- Cliquer sur "Post Comment"
- API attendue: `POST /api/articles/{slug}/comments`

3. Vérifier l'affichage du commentaire

- Commentaire apparaît immédiatement
- Informations auteur correctes (username, avatar)
- Horodatage présent
- Bouton "Delete" visible (si propriétaire)

4. Charger les commentaires existants

- Recharger la page
- API attendue : `GET /api/articles/{slug}/comments`
- Tous les commentaires s'affichent





5. Supprimer le commentaire

- Cliquer sur "Delete" du commentaire créé
- API attendue : `DELETE /api/articles/{slug}/comments/{id}`
- Confirmation de suppression

6. Vérifier la suppression

- Commentaire disparaît de l'interface
- Pas d'erreur d'affichage

Résultats attendus

-  CRUD complet des commentaires
-  Autorisation : seul l'auteur peut supprimer
-  Affichage chronologique des commentaires
-  Interface réactive (ajout/suppression temps réel)

Données de test

Json

```
{  
  "comment":{  
    "body" : "Super article ! Très instructif."  
  }  
}
```


Scénario 5 : Navigation et filtrage des articles

Objectif

Valider la page d'accueil, la pagination et les filtres d'articles

Prérequis

- Base de données avec au moins 25 articles
- Articles avec différents tags et auteurs
- Utilisateur authentifié pour tester le feed personnel

Étapes

1. Accéder à la page d'accueil

- URL : `/#/`
- Vérifier l'affichage de la liste d'articles

2. Tester la pagination

- API attendue : `GET /api/articles?limit=20&offset=0`
- Vérifier que 20 articles maximum s'affichent
- Cliquer sur "Next" pour page suivante
- Vérifier le paramètre `offset=20` dans l'URL API

3. Filtrer par tag populaire

- Section "Popular Tags" visible à droite
- Cliquer sur un tag (ex: "reactjs")
- API attendue : `GET /api/articles?tag=reactjs`
- Articles filtrés affichés avec le tag sélectionné

4. Tester le feed personnel

- Cliquer sur onglet "Your Feed"
- API attendue : `GET /api/articles/feed`
- Seuls les articles des utilisateurs suivis

5. Revenir au feed global

- Cliquer sur onglet "Global Feed"
 - API attendue : `GET /api/articles`
 - Tous les articles affichés
-

6. Filtrer par auteur

- Cliquer sur un nom d'auteur d'article
- Naviguer vers `/#/@{username}`
- API attendue : `GET /api/articles?author={username}`
- Articles de cet auteur uniquement

Résultats attendus

- ☒ Pagination fonctionnelle (20 articles/page par défaut)
- ☒ Filtres par tag appliqués correctement
- ☒ Feed personnel vs global distincts
- ☒ Navigation fluide entre les vues
- ☒ URLs mises à jour selon le contexte

Scénario 6 : Gestion de profil et relations sociales

Objectif

Tester l'affichage de profils et les fonctionnalités de follow/unfollow

Prérequis

- 2 comptes utilisateurs : follower et target_user
- target_user a publié plusieurs articles

Étapes

1. Se connecter comme follower
 - Login avec credentials follower
2. Consulter profil public
 - Naviguer vers `/#/{target_user}`
 - API attendue : `GET /api/profiles/{target_user}`
3. Vérifier les informations de profil
 - Username, bio, avatar affichés
 - Onglets "My Articles" et "Favorited Articles"
 - Bouton "Follow" visible (car pas encore suivi)
4. Suivre l'utilisateur
 - Cliquer sur "Follow {target_user}"
 - API attendue : `POST /api/profiles/{target_user}/follow`
5. Vérifier le follow
 - Code retour : 200
 - Bouton change pour "Unfollow {target_user}"
 - Profil indique `following: true`
6. Tester le feed personnel
 - Retourner à l'accueil `/#/`
 - Cliquer sur onglet "Your Feed"
 - Articles de target_user présents dans le feed
7. Unfollow l'utilisateur
 - Retourner au profil `/#/{target_user}`
 - Cliquer sur "Unfollow {target_user}"
 - API attendue : `DELETE /api/profiles/{target_user}/follow`

8. Vérifier l'unfollow

- Bouton redevient "Follow {target_user}"
- Feed personnel ne contient plus ses articles

Résultats attendus

- ☒ Profils publics accessibles sans authentification
- ☒ Follow/Unfollow impacte le feed personnel
- ☒ Boutons d'état mis à jour dynamiquement
- ☒ Relations sociales persistantes

Cas de Test Négatifs

Gestion d'erreurs API

Test : Tentative de connexion avec mot de passe incorrect

- API : **POST /api/users/login**
- Données : **`{"user":{"email":"valid@email.com","password":"wrongpassword"}}`**
- Attendu : Code 422, **`{"errors":{"email or password":["is invalid"]}}`**

Test : Création d'article sans authentification

- API : **POST /api/articles**
- Headers : Pas de token Authorization
- Attendu : Code 401, erreur d'authentification

Test : Accès à un article inexistant

- API : **GET /api/articles/article-inexistant-slug**
- Attendu : Code 404, article non trouvé

Test : Tentative de suppression d'article par non-propiétaire

- API : **DELETE /api/articles/{slug}**
- Context : Token d'un utilisateur différent de l'auteur
- Attendu : Code 403, accès refusé

Ces scénarios couvrent les parcours utilisateur principaux et les cas d'erreur importants, en s'appuyant directement sur les spécifications API RealWorld que nous avons consultées.