LU3IN017 - Technologies du Web 2021
LOI Alessia, PIRONIN Julie - Groupe 2

Projet Birdy

Tableau des services

Concept. Notre réseau social s'adressera aux passionnés de littérature.

Les utilisateurs seront des lecteurs, dont le profil présentera une courte description et un aperçu sur leurs goûts personnels concernant les livres lus, ceux à lire et d'autres informations dans cet univers. Un lecteur pourra définir sa liste d'amis et ses préférences en termes de livres et auteurs. Il peut également voir les statistiques le concernant, c'est-à-dire, le nombre de livres lus dans le mois, l'année, le nombre de commentaires (ici appelés "plumes") postés, etc.

Dans notre conception, les lecteurs, les livres et les auteurs trouveront un espace consacré : cet espace affichera les informations ainsi que l'ensemble de posts le concernant.

Chaque livre possède ainsi un profil, sur lequel les utilisateurs peuvent partager leurs ressentis sur la fin du livre, et ainsi débattre avec d'autres lecteurs qui ont terminé le livre, ou alors écrire une petite revue pour donner envie à ceux qui ne l'ont pas encore lu de le découvrir (il existe donc deux catégories distinctes).

À partir de la page/du profil d'un livre, il sera également possible de proposer ou participer à des séances de lecture en live (vidéo ou podcast) à un horaire fixé, qui sera notifié sur le site web.

On envisage d'implanter un système d'invitation ponctuelles par mail qui proposera des petits défis à relever à des horaires et échéances fixés.

Notre philosophie. Nous voudrions recréer l'atmosphère d'un cercle littéraire, avec pour chacun la possibilité de partager son expérience de lecture et d'agrandir ses horizons. Les données personnelles du lecteur seront assez masquées et surtout pas transmises à des tiers. Tout contexte commercial est banni : les auteurs et les éditeurs ne pourront pas promouvoir leurs œuvres, mais ils pourront bien évidemment comme les autres manifester leurs préférences de lecture.

La newsletter portera une attention particulière à l'utilisateur, qui sera invité à participer à des événements ou à des activités selon ses propres intérêts, dans un style amusant et convivial.

Pages HTML prévues : home, signin/login, profileUser, profileBook, profileAuthor, writePlume (= écrire un post, un tweet), messages (chat privée), friendsList, agenda, notifications, explore, settings

Légende: 🔺 : paramètre obligatoire

Authentification

Nom du WS	CreateUser
URL	birdy.com/users
Description	Création d'un nouvel utilisateur
Paramètres en entrée	▲ nom_utilisateur (login) ▲ adresse email ▲ code d'authentification de l'adresse mail ▲ mot de passe ▲ image de profil ou personnelle ou parmi celles proposées par défaut domaines d'intérêt à cocher phrase de profil : une citation
Format de sortie	JSON
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) déjà pris { "status" : "error", "message" : "id utilisé" } / Status HTTP : 409 • format de mot de passe erroné { "status" : "error", "message" : "format mdp erroné" } / Status HTTP : 406 • format de la mail erroné { "status" : "error", "message" : "format mail erroné" } / Status HTTP : 406 • code d'authentification de l'adresse email erroné { "status" : "error", "message" : "code auth. erroné" } / Status HTTP : 401 • paramètre obligatoire manquant { "status" : "error", "message" : "paramètre manquant" } / Status HTTP : 412
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	Login
URL	birdy.com/users/login
Description	Accès à un compte utilisateur
Paramètres en entrée	▲ nom_utilisateur (login) ou adresse email ▲ mot de passe
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileUser"
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • mail inexistant dans la BD

	{ “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • format de la mail erroné { “status” : ”error”, “message” : “format mail erroné” } / Status HTTP : 406 • mot de passe erroné { “status” : ”error”, “message” : “mot de passe erroné” } / Status HTTP : 401
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	ModifyPassword
URL	birdy.com/users/modify-password
Description	Modification du mot de passe
Paramètres en entrée	▲ nom_utilisateur (login) ou adresse mail ▲ code d'authentification de l’adresse mail
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok”, “password” : “R5foWdlsTKfbYt” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • mail inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • code d'authentification de l’adresse email erroné { “status” : ”error”, “message” : “code auth. erroné" } / Status HTTP : 401
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	CreateBook
URL	birdy.com/books
Description	Création d’un profil pour un livre qui n’a pas encore de profile
Paramètres en entrée	▲ titre ▲ author ▲ année de publication ▲ editeur synopsis images/illustrations citations
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• livre déjà existant dans la BD { “status” : ”error”, “message” : “livre existant” } / Status HTTP : 403

	• paramètre obligatoire manquant { “status” : ”error”, “message” : "paramètre manquant” } / Status HTTP : 412
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/books.js

Nom du WS	CreateAuthor
URL	birdy.com/authors
Description	Création d’un profil pour un auteur qui n’a pas encore de profile
Paramètres en entrée	▲ nom de l’auteur ▲ prenom de l’auteur ▲ date de naissance ▲ lieu de naissance date de décès lieu de décès images/illustrations bibliographie citations
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• auteur déjà existant dans la BD { “status” : ”error”, “message” : “author existant” } / Status HTTP : 403 • paramètre obligatoire manquant { “status” : ”error”, “message” : "paramètre manquant” } / Status HTTP : 412
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/books.js

Nom du WS	ShowMyProfile
URL	birdy.com/users/<nom_utilisateur(login)>
Description	Affichage de la page profil de l’utilisateur
Paramètres en entrée	▲ nom_utilisateur (login) ou adresse mail ▲ code d'authentification de l’adresse mail
Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok”, “password” : “R5foWdlsTKfbYt” } / Status HTTP : 200 page HTML “profileUser”
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • mail inexistant dans la BD

	{ “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • code d'authentification de l’adresse email erroné { “status” : ”error”, “message” : “code auth. erroné" } / Status HTTP : 401
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	DeleteUser
URL	birdy.com/settings/delete-account
Description	Suppression d’un compte utilisateur
Paramètres en entrée	▲ nom_utilisateur (login) ou adresse mail ▲ mot de passe
Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200 page HTML “signin/login”
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • mail inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • paramètre obligatoire manquant { “status” : ”error”, “message” : "paramètre manquant” } / Status HTTP : 412 • mot de passe erroné { “status” : ”error”, “message” : “mot de passe erroné” } / Status HTTP : 401
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/settings.js

Rendez-vous lectures

Nom du WS	PostRDVLecture
URL	birdy.com/agenda/<nom_utilisateur (login)>
Description	Créer un rendez-vous lecture
Paramètres en entrée	▲ nom_utilisateur (login)
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur inexistant dans la BD { “status” : ”error”, “message” : “nom_utilisateur inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/rdv.js

Nom du WS	PrintRDVLecture
URL	birdy.com/agenda/<nom_utilisateur (login)>
Description	Afficher un rendez-vous lecture
Paramètres en entrée	▲ nom_utilisateur (login) ▲ rdv_id
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur inexistant dans la BD { “status” : ”error”, “message” : “nom_utilisateur inexistant” } / Status HTTP : 404 • rdv_id inexistant dans la BD { “status” : ”error”, “message” : “rdv_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/rdv.js

Nom du WS	ModifyRDVLecture
URL	birdy.com/agenda/<login>/<rdv_id>
Description	Modifier les informations sur un un rendez-vous lecture
Paramètres en entrée	▲ nom_utilisateur (login) ▲ rdv_id
Format de sortie	JSON

Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur inexistant dans la BD { “status” : ”error”, “message” : “nom_utilisateur inexistant” } / Status HTTP : 404 • rdv_id inexistant dans la BD { “status” : ”error”, “message” : “rdv_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/rdv.js

Nom du WS	DeleteRDVLecture
URL	birdy.com/agenda/<nom_utilisateur (login)>/<rdv_id>
Description	Supprimer un rendez-vous lecture
Paramètres en entrée	▲ nom_utilisateur (login) ▲ rdv_id
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur inexistant dans la BD { “status” : ”error”, “message” : “nom_utilisateur inexistant” } / Status HTTP : 404 • rdv_id inexistant dans la BD { “status” : ”error”, “message” : “rdv_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/rdv.js

Messages

Nom du WS	SendMessage
URL	birdy.com/messages
Description	Envoyer un message à un ami.
Paramètres en entrée	▲ nom_utilisateur (login) de l'ami ou id_utilisateur de l'ami
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "messages"
Erreurs possibles	• nom_utilisateur (login) de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • id_utilisateur de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/messages.js

Nom du WS	PrintConversation
URL	birdy.com/messages/<nom_ami>-<nom_utilisateur(login)>
Description	Voir la discussion avec un ami
Paramètres en entrée	▲ nom_utilisateur (login) de l'ami ou id_utilisateur de l'ami ▲ nom_utilisateur (login) ou adresse mail
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "messages"
Erreurs possibles	• nom_utilisateur (login) de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • id_utilisateur de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/messages.js

Nom du WS	DeleteConversation
URL	birdy.com/messages/<nom_ami>-<nom_utilisateur(login>
Description	Supprimer la discussion avec un ami

Paramètres en entrée	▲ nom_utilisateur (login) de l'ami ou id_utilisateur de l'ami ▲ nom_utilisateur (login) ou adresse mail
Format de sortie	JSON
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • id_utilisateur de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/messages.js

Amis

Nom du WS	AddFriend
URL	birdy.com/users/<nom_ami(login)>
Description	Ajouter un ami
Paramètres en entrée	▲ nom_utilisateur (login) de l'ami ou id_utilisateur de l'ami
Format de sortie	JSON
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • id_utilisateur de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	ShowFriendProfile
URL	birdy.com/users/<nom_ami(login)>
Description	Affichage de la page profil de l'ami
Paramètres en entrée	▲ nom_utilisateur (login) de l'ami ou id_utilisateur de l'ami
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileUser"
Erreurs possibles	• nom_utilisateur (login) de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404 • id_utilisateur de l'ami inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	ShowFriendsList
URL	birdy.com/users/<friends-list>
Description	Affichage de la liste d'amis
Paramètres en entrée	▲ nom_utilisateur (login)

Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200 page HTML “friendsList”
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • liste d’amis vide { “status” : ”error”, “message” : “friends list empty” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Nom du WS	DeleteFriend
URL	birdy.com/users/<friends-list>
Description	Suppression d’un ami depuis la liste d’amis
Paramètres en entrée	▲ id_utilisateur de l’ami ▲ nom_utilisateur (login)
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “login inexistant” } / Status HTTP : 404 • id_utilisateur ami (login) inexistant dans la BD { “status” : ”error”, “message” : “id_utilisateur ami inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/users.js

Recherche

Nom du WS	SearchAuthor
URL	birdy.com/explore/search?q=<nom ou prenom auteur>&src=typed_query
Description	Rechercher un auteur
Paramètres en entrée	▲ nom de l'auteur ou prénom de l'auteur titre
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileAuthor"
Erreurs possibles	• nom_auteur (login) inexistant dans la BD { "status" : "error", "message" : "nom_auteur inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/explore.js

Nom du WS	SearchBook
URL	birdy.com/explore/search?q=<titre>&src=typed_query
Description	Rechercher un livre
Paramètres en entrée	▲ titre author ou année de publication ou éditeur
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileBook"
Erreurs possibles	• livre inexistant dans la BD { "status" : "error", "message" : "nom_livre inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/explore.js

Nom du WS	SearchPlume
URL	birdy.com/explore/search?q=<mot>&src=typed_query
Description	Rechercher une plume
Paramètres en entrée	▲ mot présent dans la plume nom_utilisateur (login)

Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• mot inexistant dans la BD (aucune plume contenant ce mot) { “status” : ”error”, “message” : “plume inexistante” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/explore.js

Nom du WS	SearchUser
URL	birdy.com/explore/search?q=<nom_utilisateur>&src=typed_query
Description	Rechercher un utilisateur
Paramètres en entrée	🔺 nom_utilisateur (login)
Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200 page HTML “profileUser”
Erreurs possibles	• nom_user (login) inexistant dans la BD { “status” : ”error”, “message” : “nom_user inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/explore.js

Nom du WS	SearchRDV
URL	birdy.com/explore/search?q=<RDV>&src=typed_query
Description	Rechercher un rendez-vous lecture.
Paramètres en entrée	🔺 rdv_id
Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200 page HTML “agenda”
Erreurs possibles	• rdv_id (login) inexistant dans la BD { “status” : ”error”, “message” : “rdv_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/explore.js

Commentaires

Nom du WS	WriteCommentaire
URL	birdy.com/authors/<author_id>/avis OU birdy.com/books/<book_id>/avis OU birdy.com/books/<book_id>/envie
Description	Écrire un avis/envie sur une page d'auteur/une page de livre.
Paramètres en entrée	▲ author_id ou book_id
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileAuthor" ou "profileBook"
Erreurs possibles	• author_id (login) inexistant dans la BD { "status" : "error", "message" : "author_id inexistant" } / Status HTTP : 404 • book_id (login) inexistant dans la BD { "status" : "error", "message" : "book_id inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/commentaire.js

Nom du WS	DeleteCommentaire
URL	birdy.com/authors/<author_id>/avis/commentaire_id OU birdy.com/books/<book_id>/avis/commentaire_id OU birdy.com/books/<book_id>/envie/commentaire_id
Description	Supprimer un résumé (donne envie de lire le livre) sur une page de livre
Paramètres en entrée	▲ nom utilisateur (login) ▲ commentaire_id
Format de sortie	JSON
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200
Erreurs possibles	• envie_id (résumé) inexistant dans la BD { "status" : "error", "message" : "plume inexistante" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/commentaire.js

Nom du WS	PrintCommentaire
URL	birdy.com/authors/<author_id>/avis OU birdy.com/books/<book_id>/avis OU

	birdy.com/books/<book_id>/envie
Description	Afficher les commentaires (avis/envie) sur une page d'auteur/livre.
Paramètres en entrée	🔺 author_id ou book_id
Format de sortie	JSON HTML
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200 page HTML "profileBook" ou page HTML "profileAuthor"
Erreurs possibles	• author_id (login) inexistant dans la BD { "status" : "error", "message" : "author_id inexistant" } / Status HTTP : 404 • book_id (login) inexistant dans la BD { "status" : "error", "message" : "book_id inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/commentaire.js

Accueil

Nom du WS	Home
URL	birdy.com/home
Description	Afficher le fil d'actualité général
Paramètres en entrée	🔺 nom_utilisateur (login)
Format de sortie	JSON
Exemple de sortie	{ "status" : "ok" } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { "status" : "error", "message" : "login inexistant" } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/home.js

Plumes

Nom du WS	PostPlume
URL	birdy.com/compose/plume
Description	Écrire une plume (une sorte de tweet).
Paramètres en entrée	🔺 nom_utilisateur (login)
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• nom_utilisateur (login) inexistant dans la BD { “status” : ”error”, “message” : “nom_utilisateur inexistant” } / Status HTTP : 404 • format du message plume erroné { “status” : ”error”, “message” : “format erroné” } / Status HTTP : 406
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/plume.js

Nom du WS	DeletePlume
URL	birdy.com/<nom_utilisateur(login)>/plumes/<plume_id>
Description	Écrire un avis sur une page d’auteur
Paramètres en entrée	🔺 plume_id
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200
Erreurs possibles	• plume_id inexistant dans la BD { “status” : ”error”, “message” : “plume_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/plume.js

Nom du WS	PrintPlume
URL	birdy.com/<nom_utilisateur(login)>/plumes
Description	Affiche les plumes écrites par un utilisateur
Paramètres en entrée	🔺 plume_id
Format de sortie	JSON
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200

Erreurs possibles	• plume_id inexistant dans la BD { “status” : ”error”, “message” : “plume_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/plume.js

Notifications

Nom du WS	PrintNotification
URL	birdy.com/notifications
Description	Affiche les notifications (liées aux amis, plumes, ou rendez-vous littéraires)
Paramètres en entrée	🔺 plume_id
Format de sortie	JSON HTML
Exemple de sortie	{ “status” : ”ok” } / Status HTTP : 200 page HTML “Notifications”
Erreurs possibles	• plume_id inexistant dans la BD { “status” : ”error”, “message” : “plume_id inexistant” } / Status HTTP : 404
Classes/Fichiers JS	src/index.js ; src/api.js ; src/entities/notifications.js