Total: **48 / 51**

Réalisé ?	Fichier	Catégorie	Nom du test	Route API	Type de test	Données en entrée	Résultat attendu	Auteur du test
☑	algos.test.ts	GET /api/algos/byUserId/:id	erreur: Aucun algorithme trouvé.	/api/algos/byUserId/3	Echec	- Token authentifié et valide - Id utilisateur: 3	- Statut de retour: 404 - Message: "Aucun algorithme trouvé."	Mathieu
☑	algos.test.ts	GET /api/algos/byUserId/:id	succès: Algorithmes trouvés.	/api/algos/byUserId/110	Succès	- Token authentifié et valide - Id utilisateur: 110	- Statut de retour: 200 - Message: "Algorithmes trouvés." - Data: nombre d'algos == 1	Mathieu
☑	algos.test.ts	GET /api/algos/:id	erreur: Algorithme non trouvé.	/api/algos/13	Echec	- Token authentifié et valide - Id algorithme: 13	- Statut de retour: 404 - Message: "Algorithme non trouvé."	Mathieu
☑	algos.test.ts	GET /api/algos/:id	succès: Algorithme trouvé.	/api/algos/1	Succès	- Token authentifié et valide - Id algorithme: 1	- Statut de retour: 201 - Message: "Algorithme trouvé." - Données: métadonnées + algo-complet.json	Mathieu
☑	algos.test.ts	POST /api/algos/:id	erreur: Données manquantes.	/api/algos	Echec	- Token authentifié et valide	- Statut de retour: 400 - Message: "Il manque des données."	Mathieu
☑	algos.test.ts	POST /api/algos/:id	erreur: Algorithme invalide.	/api/algos	Echec	- Token authentifié et valide - Nom: "Algorithme test" - Id propriétaire: 110 - Code source: algo-3.json	- Statut de retour: 400 - Message: "Algorithme invalide." - Données: objet indiquant l'erreur trouvé. - Pas de fichier stocké dans l'application.	Mathieu
☑	algos.test.ts	POST /api/algos/:id	succès: Algorithme créé.	/api/algos	Succès	- Token authentifié et valide - Nom: "Algorithme test" - Id propriétaire: 110 - Code source: algo-complet.json	- Statut de retour: 201 - Message: "Algorithme créé." - Données: métadonnées - Fichier stocké dans l'application.	Mathieu
☑	algos.test.ts	PUT /api/algos/:id	erreur: Données manquantes.	/api/algos/1	Echec	- Token authentifié et valide	- Statut de retour: 400 - Message: "Il manque des données."	Mathieu
☐	algos.test.ts	PUT /api/algos/:id	erreur: Algorithme invalide.	/api/algos/1	Echec	- Token authentifié et valide - Nom: "Algorithme test mise à jour" - Id propriétaire: 110 - Code source: algo-3.json	- Statut de retour: 400 - Message: "Algorithme invalide." - Données: objet indiquant l'erreur trouvé. - Algorithme stocké non mise à jour.	Mathieu
☑	algos.test.ts	PUT /api/algos/:id	succès: Algorithme créé.	/api/algos/1	Succès	- Token authentifié et valide - Nom: "Algorithme test mise à jour" - Id propriétaire: 110 - Code source: algo-1.json	- Statut de retour: 201 - Message: "Algorithme mise à jour." - Données: métadonnées - Fichier mise à jour dans l'application.	Mathieu
☑	algos.test.ts	DELETE /api/algos/:id	erreur: Algorithme non trouvé.	/api/algos/13	Echec	- Token authentifié et valide - Id algorithme: 13	- Statut de retour: 404 - Message: "Algorithme non trouvé."	Mathieu
☑	algos.test.ts	DELETE /api/algos/:id	succès: Algorithme trouvé.	/api/algos/1	Succès	- Token authentifié et valide - Id algorithme: 1	- Statut de retour: 201 - Message: "Algorithme supprimé." - Fichier stocké bien supprimé dans l'application.	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°1. -> aucune erreur: présence que de problèmes.		Succès	- Algorithme: algo-1.json	- Objet dont la propriété success == true	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°2. -> manquant: listeDonnes, listeResultats.		Succès	- Algorithme: algo-2.json	- Objet dont la propriété success == true	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°3. -> 8 erreurs: mauvais format de abscisse/ordonnee.		Echec	- Algorithme: algo-3.json	Objet: - dont la propriété success == false - dont la propriété errors contient 8 items.	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°4. -> aucune erreur: présence de décompositions.		Succès	- Algorithme: algo-4.json	- Objet dont la propriété success == true	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°5. -> aucune erreur: présence de switch.		Succès	- Algorithme: algo-5.json	- Objet dont la propriété success == true	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°6. -> aucune erreur: Dictionnaire vide.		Succès	- Algorithme: algo-6.json	- Objet dont la propriété success == true	Mathieu
☑	algos.test.ts	Algos: Validator	Algo référence n°7. -> aucune erreur: Dictionnaire valide.		Succès	- Algorithme: algo-7.json	- Objet dont la propriété success == true	Jokin
☑	algos.test.ts	Algos: Validator	Algo référence n°8. -> 2 erreurs: Dictionnaire avec types de données invalides.		Echec	- Algorithme: algo-8.json	Objet: - dont la propriété success == false - dont la propriété errors contient 2 items.	Jokin
☑	algos.test.ts	Algos: Validator	Algo référence n°10. -> aucune erreur.		Succès	- Algorithme: algo-complet.json	- Objet dont la propriété success == true	Mathieu
☑	users.test.ts	POST /api/users/register	erreur: Il manque des données.	/api/users/register	Echec	Rien	- Statut de retour: 400 - Message: "Il manque des données."	Mathieu
☑	users.test.ts	POST /api/users/register	erreur: Email invalide.	/api/users/register	Echec	- Email: "test@@" - Mot de passe: "testtest" - pseudo: "Test de ToxykAuBleu"	- Statut de retour: 400 - Message: "Données invalides." - Données: objet dont la propriété propriété email existe.	Mathieu
☑	users.test.ts	POST /api/users/register	erreur: Mot de passe trop court.	/api/users/register	Echec	- Email: "test@toxykaubleu.fr" - Mot de passe: "test" - pseudo: "Test de ToxykAuBleu"	- Statut de retour: 400 - Message: "Données invalides." - Données: objet dont la propriété propriété password existe.	Mathieu

☑	users.test.ts	POST /api/users/register	succès: Utilisateur créé.	/api/users/register	Succès	- Email: "test@toxykaubleu.fr" - Mot de passe: "testtest" - pseudo: "Test de ToxykAuBleu"	- Statut de retour: 201 - Message: "Utilisateur créé." - Email envoyé: - dont le destinataire est l'email en entrée; - dont le sujet est "Confirmation mail"; - dont le contenu du mail contient le lien de vérification.	Mathieu
☑	users.test.ts	POST /api/users/register	erreur: Email déjà utilisé.	/api/users/register	Echec	- Email: "test@toxykaubleu.fr" - Mot de passe: "testtest" - pseudo: "Test de ToxykAuBleu"	- Statut de retour: 409 - Message: "Email déjà utilisé."	Mathieu
☑	users.test.ts	GET /api/users/confirm/:token	erreur: Token invalide. (wrong)	/api/users/confirm/wrong	Echec	- Token de confirmation: wrong	- Statut de retour: 400 - Message: "Token invalide."	Mathieu
☑	users.test.ts	GET /api/users/confirm/:token	erreur: Token invalide. (Ex: base64)	/api/users/confirm/MTEyX1VGVDE3MzYyNjQ5ODk4ODE=	Echec	- Token de confirmation: MTEyX1VGVDE3MzYyNjQ5ODk4ODE=	- Statut de retour: 400 - Message: "Token invalide."	Mathieu
☑	users.test.ts	GET /api/users/confirm/:token	succès: Inscription confirmée.	/api/users/confirm/:token	Succès	- Token de confirmation: réutiliser celui créé lors de l'inscription.	- Statut de retour: 200 - Message: "Inscription confirmée."	Mathieu
☑	users.test.ts	POST /api/users/login	erreur: Il manque des données.	/api/users/login	Echec	Rien	- Statut de retour: 400 - Message: "Il manque des données."	Mathieu
☑	users.test.ts	POST /api/users/login	erreur: Utilisateur introuvable.	/api/users/login	Echec	- Email: "test@example.com" - Mot de passe: "testtest"	- Statut de retour: 404 - Message: "Utilisateur introuvable."	Mathieu
☑	users.test.ts	POST /api/users/login	erreur: Mot de passe incorrect.	/api/users/login	Echec	- Email: "test@toxykaubleu.fr" - Mot de passe: "wrong"	- Statut de retour: 400 - Message: "Mot de passe incorrect."	Mathieu
☑	users.test.ts	POST /api/users/login	succès: Connexion réussie.	/api/users/login	Succès	- Email: "test@toxykaubleu.fr" - Mot de passe: "testtest"	- Statut de retour: 200 - Message: "Connexion réussie." - Header de la réponse contient le token à utiliser partout.	Mathieu
☑	users.test.ts	GET /api/users/logout	erreur: Token manquant.	/api/users/logout	Echec	Rien	- Statut de retour: 400 - Message: "Token manquant."	Mathieu
☑	users.test.ts	GET /api/users/logout	succès: Déconnexion réussie.	/api/users/logout	Succès	- Token authentifié et valide	- Statut de retour: 200 - Message: "Déconnexion réussie."	Mathieu
☑	users.test.ts	GET /api/users/:id	erreur: Token manquant.	/api/users/1	Echec	- Id utilisateur: 1	- Statut de retour: 400 - Message: "Token manquant."	Mathieu
☑	users.test.ts	GET /api/users/:id	erreur: Token invalide.	/api/users/1	Echec	- Token authentifié et invalide - Id utilisateur: 1	- Statut de retour: 401 - Message: "Token invalide."	Mathieu
☑	users.test.ts	GET /api/users/:id	erreur: Token invalide.	/api/users/112	Succès	- Token authentifié et valide - Id utilisateur: 112	- Statut de retour: 200 - Message: "Utilisateur trouvé."	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 40 -> erreur: Permission refusée.	/api/users/40	Echec	- Token authentifié et valide - Connecté avec l'utilisateur 112	- Statut de retour: 401 - Message: "Permission refusée."	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 112 -> erreur: Mot de passe incorrect.	/api/users/112	Echec	- Token authentifié et valide - Connecté avec l'utilisateur 112 - Pseudo: "Nouveau pseudo !" - Mot de passe actuel: "wrong"	- Statut de retour: 401 - Message: "Mot de passe incorrect."	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 112 -> erreur: Url de la photo de profil invalide.	/api/users/112	Echec	- Token authentifié et valide - Connecté avec l'utilisateur 112 - Url photo de profil: "wrong" - Mot de passe actuel: "testtest"	- Statut de retour: 400 - Message: "Données invalides." - Données: objet dont la propriété urlPfp existe.	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 112 -> erreur: Url de la photo de profil n'est pas une image.	/api/users/112	Echec	- Token authentifié et valide - Connecté avec l'utilisateur 112 - Url photo de profil: "http://google.com" - Mot de passe actuel: "testtest"	- Statut de retour: 400 - Message: "URL de photo de profil invalide."	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 112 -> succès: Url de la photo de profil modifié.	/api/users/112	Succès	- Token authentifié et valide - Connecté avec l'utilisateur 112 - Url photo de profil valide - Mot de passe actuel: "testtest"	- Statut de retour: 200 - Message: "Utilisateur mis à jour."	Mathieu
☑	users.test.ts	PUT /api/users/:id	ID: 112 -> succès: Mot de passe modifié.	/api/users/112	Succès	- Token authentifié et valide - Connecté avec l'utilisateur 112 - Mot de passe actuel: "testtest" - Nouveau mot de passe: "nouveauMdp"	- Statut de retour: 200 - Message: "Utilisateur mis à jour."	Mathieu
☑	users.test.ts	DELTE /api/users/:id	ID: 101 -> erreur: Token invalide.	/api/users/101	Echec	- Token authentifié et invalide. - Id utilisateur: 101	- Statut de retour: 401 - Message: "Token invalide."	Mathieu
☑	users.test.ts	DELTE /api/users/:id	ID: 1 -> erreur: Utilisateur introuvable.	/api/users/1	Echec	- Token authentifié et valide. - Connecté avec l'utilisateur 101 - Id utilisateur: 1	- Statut de retour: 404 - Message: "Utilisateur introuvable."	Mathieu
☑	users.test.ts	DELTE /api/users/:id	ID: 102 -> erreur: Permission refusée. (Connecté: User (ID:101))	/api/users/102	Echec	- Token authentifié et valide. - Connecté avec l'utilisateur 101 - Id utilisateur: 102	- Statut de retour: 403 - Message: "Permission refusée."	Mathieu
☑	users.test.ts	DELTE /api/users/:id	ID: 102 -> succès: Utilisateur supprimé.	/api/users/102	Succès	- Token authentifié et valide. - Connecté avec l'utilisateur 102 - Id utilisateur: 102	- Statut de retour: 200 - Message: "Utilisateur supprimé."	Mathieu

☑	users.test.ts	GET /api/users/quota	succès: Quota de l'utilisateur trouvé.	/api/users/quota	Succès	- Token authentifié et valide.	- Statut de retour: 200 - Message: "Quota de l'utilisateur trouvé." - Données: objet dont used == 0 et max == 500	Mathieu
☐	users.test.ts	POST /api/users/recover	erreur: Email introuvable.	/api/users/recover	Succès	- Email: "test@example.com"	- Statut de retour: 200 - Message: "Mail de récupération envoyé."	Mathieu
☐	users.test.ts	POST /api/users/recover	succès: Mail de récupération envoyé.	/api/users/recover	Succès	- Email: "test@toxykaubleu.fr"	- Statut de retour: 200 - Message: "Mail de récupération envoyé."	Mathieu