Documentation technique

- **Nom de l'application** : Sportify
- **Description** : Application de suivi sportif permettant aux utilisateurs de créer des exercices, des séances, et de suivre leur poids/taille.
- **Objectif** : Offrir un outil personnalisé pour le suivi de la progression sportive.
- **Public visé** : Utilisateurs individuels souhaitant suivre leurs entraînements et mensurations.

Stack technique

- NextJS (React) ⇒ Javascript
- TailwindCSS
- API REST de NextJS
- Base de données PostgreSQL
- Système d’authentification
  - Hachage bcrypt des mots de passe
  - Cookie JWT dans le navigateur

Architecture de l’application

- /src
  - /app
    - (app) ⇒ Pages de l’application
    - (auth) ⇒ Pages d’authentification
    - api ⇒ Routes API

        - page.jsx ⇒ Landing Page
    - /components ⇒ Composants JSX réutilisables
    - /hooks ⇒ Hooks React personnalisés
    - /libs ⇒ Fonctions utilitaires de différentes librairies
    - /util ⇒ Fonctions utilitaires
    - middleware.js ⇒ Paramétrage du middleware de l'application

Fonctionnalités

Authentification

- Création de compte
- Connexion / Déconnexion
- Réinitialisation du mot de passe

Gestion des exercices

- CRUD : création, modification, suppression
- Possibilité de rendre un exercice public

Gestion des séances

- Création, modification et suppression de séances contenant des exercices
- Affichage sous forme de planning

Suivi des mensurations

- Saisie du poids et de la taille à une date donnée

API

Authentification

Connexion

Point d'accès: `POST /api/auth/login`

Authentifie un utilisateur et crée une session.

Corps de la requête:

```
{
  "userLogin": "nom_utilisateur",
  "password": "mot_de_passe"
}
```

Réponses:

- `201 Created`: Connexion réussie

  ```
  {
    "message": "Connexion réussie",
    "token": "jeton_de_session"
  }
  ```

- `400 Bad Request`: Données de formulaire invalides
- `401 Unauthorized`: Nom d'utilisateur ou mot de passe incorrect
- `500 Internal Server Error`: Erreur serveur

Inscription

Point d'accès: `POST /api/auth/register`

Crée un nouveau compte utilisateur.

Corps de la requête:

```
{
  "pseudo": "nom_utilisateur",
  "firstname": "Prénom",
  "lastname": "Nom",
  "email": "utilisateur@exemple.com",
  "password": "mot_de_passe_sécurisé",
  "birthday": "AAAA-MM-JJ",
  "sex": "M/F/A"
}
```

Réponses:

- `201 Created` : Utilisateur créé avec succès
- `400 Bad Request` : Données de formulaire invalides
- `409 Conflict` : Nom d'utilisateur ou email déjà existant
- `500 Internal Server Error` : Erreur serveur

Gestion de session

Point d'accès: `GET /api/auth/session`

Récupère la session utilisateur courante.

Réponses:

- `200 OK` : Session trouvée

```
{
  "userId": "id_utilisateur"
}
```

- `404 Not Found` : Aucune session trouvée
- `401 Unauthorized` : Session invalide

Point d'accès: `DELETE /api/auth/session`

Déconnecte l'utilisateur en supprimant sa session.

Réponses:

- `200 OK` : Session supprimée avec succès
- `404 Not Found` : Session non trouvée ou erreur pendant la suppression

Réinitialisation de mot de passe

Point d'accès: `POST /api/auth/password`

Initie le processus de réinitialisation de mot de passe en envoyant un lien de réinitialisation à l'email de l'utilisateur.

Corps de la requête:

```
{
  "email": "utilisateur@exemple.com"
```

```
}
```

Réponses:

- `201 Created` : Clé de réinitialisation créée avec succès
- `400 Bad Request` : Données de formulaire invalides
- `404 Not Found` : Email non trouvé
- `500 Internal Server Error` : Erreur serveur

Point d'accès: `GET /api/auth/password?key={clé_de_réinitialisation}`

Vérifie la clé de réinitialisation de mot de passe.

Paramètres de requête:

- `key` : La clé de réinitialisation de mot de passe

Réponses:

- `201 Authorized` : Clé valide
- `400 Bad Request` : Aucune clé fournie
- `404 Not Found` : Clé invalide
- `500 Internal Server Error` : Erreur serveur

Utilisateur

Point d'accès: `GET /api/user`

Récupère les informations de l'utilisateur courant.

Réponses:

- `200 OK` : Données utilisateur retournées
- `500 Internal Server Error` : Erreur lors du chargement des données utilisateur

Exercices

Point d'accès: `POST /api/exercices`

Crée un nouvel exercice.

Corps de la requête:

```
{
  "title": "Nom de l'exercice",
  "description": "Description de l'exercice",
  "isPublic": true
}
```

Réponses:

- `200 OK` : Exercice créé avec succès
- `400 Bad Request` : Données invalides
- `401 Unauthorized` : Utilisateur non authentifié
- `500 Internal Server Error` : Erreur serveur

Point d'accès: `PUT /api/exercices/{id}`

Met à jour un exercice existant.

Paramètres de chemin:

- `id` : L'identifiant de l'exercice

Corps de la requête:

```
{
  "title": "Nom de l'exercice mis à jour",
  "description": "Description de l'exercice mise à jour",
  "isPublic": true
}
```

Réponses:

- `200 OK` : Exercice mis à jour avec succès
- `400 Bad Request` : Données invalides
- `500 Internal Server Error` : Erreur serveur

Point d'accès: `DELETE /api/exercices/{id}`

Supprime un exercice.

Paramètres de chemin:

- `id` : L'identifiant de l'exercice

Réponses:

- `200 OK`: Exercice supprimé avec succès
- `500 Internal Server Error`: Erreur serveur

Séances d'entraînement

Point d'accès: `POST /api/seances`

Crée une nouvelle séance d'entraînement.

Corps de la requête:

```
{
  "title": "Nom de la séance",
  "date": "AAAA-MM-JJ",
  "duration": 60,
  "objective": "Objectif de la séance",
  "exercises": [
    {
      "id_exercice": 1,
      "sets": 3,
      "reps": 10,
      "weight": 50
    }
  ]
}
```

Réponses:

- `200 OK`: Séance créée avec succès
- `400 Bad Request`: Données invalides
- `401 Unauthorized`: Utilisateur non authentifié
- `500 Internal Server Error`: Erreur serveur

Point d'accès: `GET /api/seances`

Récupère toutes les séances d'entraînement pour l'utilisateur courant.

Réponses:

- `200 OK`: Séances récupérées avec succès

- `401 Unauthorized` : Utilisateur non authentifié
- `500 Internal Server Error` : Erreur serveur

Données physiques

Point d'accès: `POST /api/physical-data`

Crée une nouvelle entrée de données physiques.

Corps de la requête:

```
{
  "taille": 180,
  "poids": 75
}
```

Réponses:

- `200 OK` : Données physiques créées avec succès
- `400 Bad Request` : Données invalides
- `401 Unauthorized` : Utilisateur non authentifié
- `500 Internal Server Error` : Erreur serveur

Point d'accès: `GET /api/physical-data`

Récupère toutes les entrées de données physiques pour l'utilisateur courant.

Réponses:

- `200 OK` : Données physiques récupérées avec succès
- `401 Unauthorized` : Utilisateur non authentifié
- `500 Internal Server Error` : Erreur serveur

Installation du projet :

```
git clone https://github.com/nRayen/Projet-Leger-E6.git
cd Projet-Leger-E6/Code
```

```
npm i
```

Créer fichier .env à la racine avec les variables d’environnement suivantes :

```
DATABASE_URL="..." ## URL de votre base de données
SESSION_SECRET="..." ## Clé secrète pour le hashage des mots de passes
RESEND_API_KEY="..."  ## Clé API de Resend pour activer l'envoi mails
NEXT_PUBLIC_API_URL="..." ## URL de l'API ou http://localhost:3000
```

Pour lancer le projet :

```
npm run dev
```

Le serveur est lancé à http://localhost:3000/