



# R-Training

**Présenté par Ryan Ladmia  
Pour le Titre de Développeur web et web mobile**

# À Propos de moi :

## Mon Parcours

- Bac L - 2015
- Faculté de droit - 2015 à 2019
- Formation CDPI La Plateforme\_ - 2024 à 2025

## Pourquoi ?

- Passion depuis l'enfance
- Intérêt plus intense depuis la formation
- Volonté de poursuivre en Cybersécurité

# Sommaire :

I - Introduction et contexte

II - Cahier des charges

III - Conception et design

IV - Architecture technique

V - Développement et fonctions clés

VI - Tests et déploiement

VII - Démonstration

VIII - Bilan personnel

Conclusion

# I - Introduction et contexte

## Présentation du projet

**Application web et web mobile de création,  
gestion et suivi de programme sportif.**

### **Pourquoi ce projet ?**

- **Centraliser les bonnes informations**
- **Fournir une application web sur navigateur plutôt que stockée sur smartphone**
- **Permettre la consultation des performances sous forme de graphiques**

# Objectifs

**Fournir un outil simple :**

- **centralise l'information**
- **accompagner l'utilisateur**
- **suivi de progression**
- **stimuler la motivation**
- **offrir une expérience personnalisée**

## II - Cahier des charges

### Public cible

R-Training s'adresse à un large public

Pratiquants débutants :

- cadre clair, conseils fiables et accompagnement

Les pratiquants intermédiaires ou confirmés :

- optimiser leurs performances et suivre leurs progression

Entraîneurs :

- Partage de connaissances et d'expérience
- Édition d'exercices et programmes fiables

Administrateurs :

- Gestion de la plateforme

# Fonctionnalités principales

## Utilisateurs :

- Sélection de programme public
- Création de programme personnel
- Suivi d'entraînement
- Visualisation des performances

## Entraîneurs :

- Cratio de programmes publics
- Partage d'exercices de musculation

## Administrateurs :

- Gestion des utilisateurs

# Contraintes techniques

- Le projet doit être développé avec React et Node.js
- Utilisation de framework Hono
- Appels API avec Axios
- Utilisation obligatoire d'un ORM (ex : Prisma)
- Base de données relationnelle PostgreSQL
- Compatibilité avec mobile (responsive design)
- Ne pas utiliser de cookies non sécurisés (sécurité HTTPS)
- Utiliser JWT et Bcrypt pour l'authentification sécurisée
- Hébergement des images avec Cloudinary

# Contraintes fonctionnelles

- **Un utilisateur peut créer un compte et se connecter.**
- **Un administrateur peut gérer les utilisateurs.**
- **L'utilisateur doit pouvoir ajouter une photo de profil.**
- **L'authentification doit se faire avec JWT.**
- **Les rôles doivent limiter l'accès à certaines pages.**
- **Les utilisateurs non connectés ne peuvent pas voir les données privées.**
- **Le site doit envoyer un e-mail de confirmation.**
- **Suivi des entraînements**
- **Visualisation des performances**
- **Ajout de consultation des exercices de musculation**

# Benchmark rapide

**PHP vs Javascript = JavaScript :**

- **Dynamique (manipule le DOM)**
- **Asynchrone**
- **Interactif**

**Symfony vs React + Node.js = React + Node.js :**

- **Flexible**
- **Vaste écosystème**
- **Interactivité côté front**

**Express vs Hono = Hono :**

- **Léger et rapide**
- **Moderne**

**PostgreSQL vs SQL vs NoSQL = PostgreSQL :**

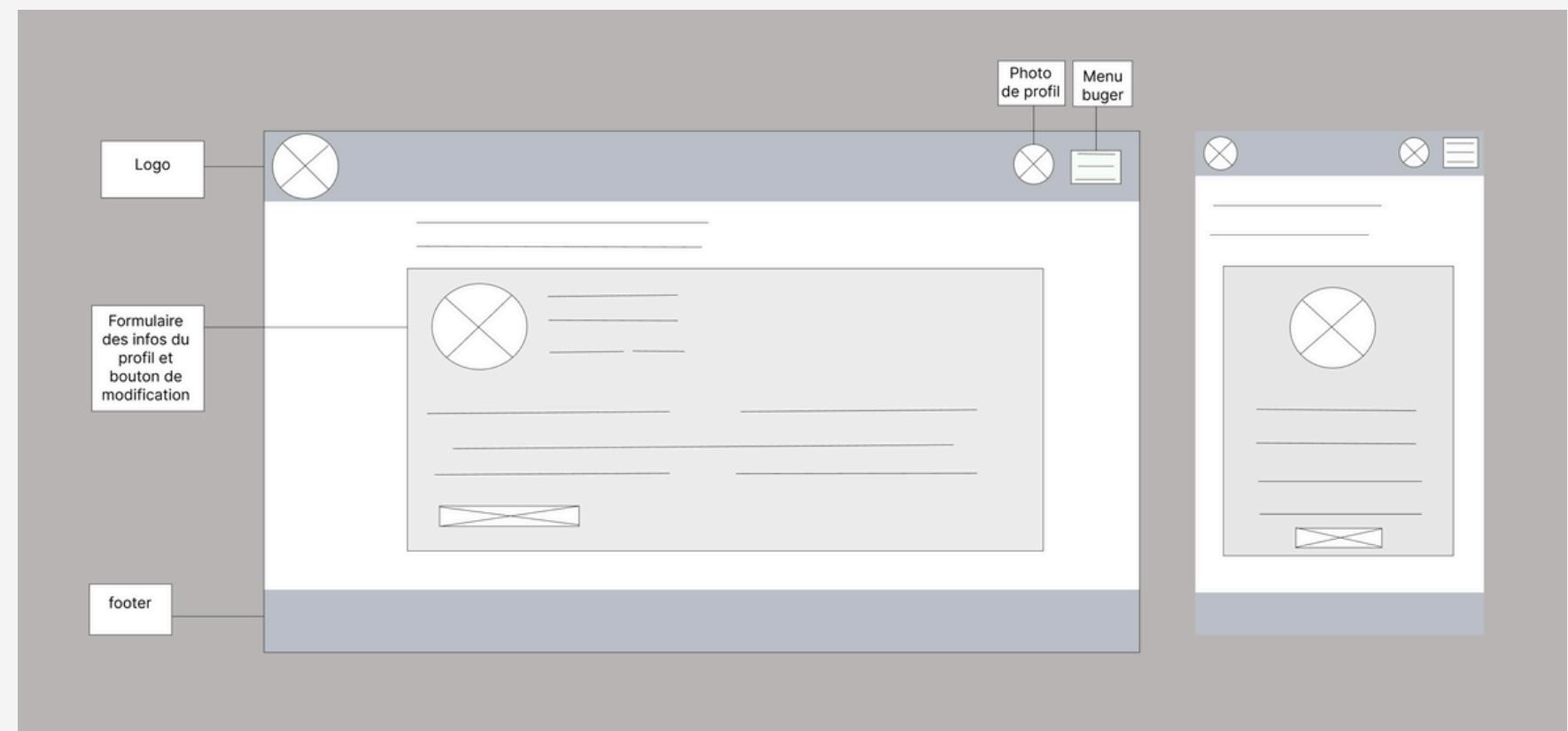
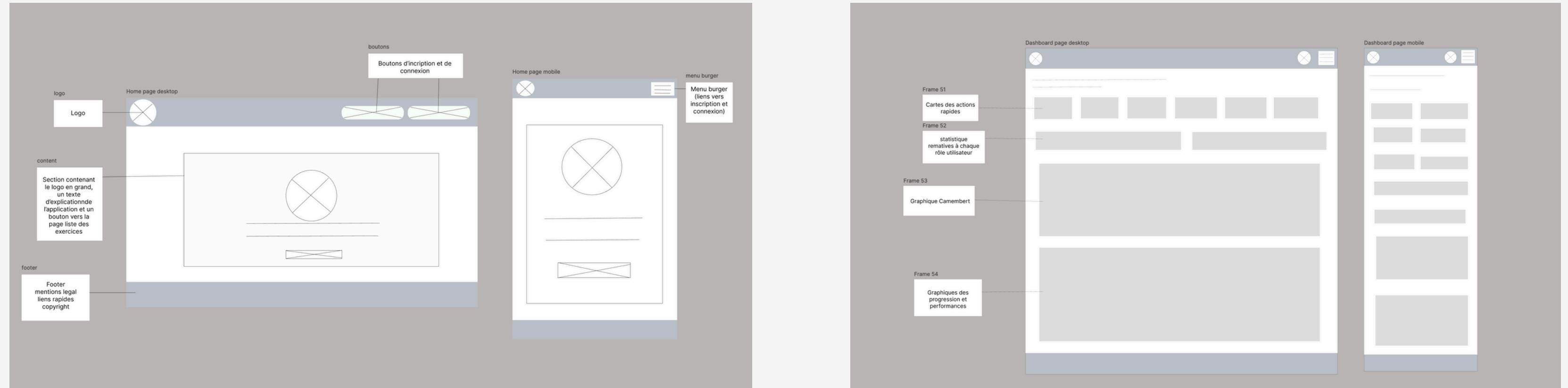
- **Excellent avec Prisma**
- **Stable, moderne et performant**
- **Requêtes sécurisées**

# III - Conception et design

## Arborescence



# Wireframes



# Maquettes

The image displays four wireframe mockups of the R-Training platform interface, arranged in a grid:

- Home page desktop:** Shows the main landing page with the R-Training logo, the tagline "Transformer votre entraînement", and a "Découvrez nos exercices" button.
- Home page mobile:** Shows the mobile version of the landing page with the same branding and call-to-action.
- Profile page desktop:** Shows the administrator profile page with fields for first name, last name, email, phone number, and birth date, along with a "Modifier le profil" button.
- Profile page mobile:** Shows the user profile page with similar fields and a "Modifier le profil" button.

The image displays two wireframe mockups of the R-Training exercise list page:

- Exercises list page desktop:** Shows three exercises: Face pull, Squat, and Soulevé de terre jambes tendues. Each exercise card includes a thumbnail, a title, a description, the muscle group (e.g., Dos, Jambes), and target muscles (e.g., Muscles ciblés).
- Exercises list page mobile:** Shows the same three exercises in a mobile-friendly layout, with each exercise card including a thumbnail, a title, a description, the muscle group, and target muscles.

# Charte graphique

Charte Graphique



Fond

Couleur Principale

Couleur Secondaire

Couleur Tertiaire

Police : Inter, system-ui sans serif

# Responsivité

**Conçu pour les différents types d'écrans et les différentes tailles :**

- Mobile S
- Mobile M
- Mobile L
- Tablette
- Laptop (ordinateur)
- Laptop L
- Grand écran 4K

# Accessibilité

L'accessibilité vise à rendre un site utilisable par tous. Cela implique des pratiques comme un contraste suffisant des couleurs ou une navigation clavier fluide.

L'UX, ou expérience utilisateur : simplicité, logique, efficacité, et satisfaction. Une bonne UX permet à l'utilisateur d'atteindre facilement son objectif. Accessibilité et UX sont donc complémentaires : l'une assure l'inclusion, l'autre améliore le confort d'usage.

# Ergonomie mobile/desktop

**L'interface s'adapte à son support. Sur desktop, on dispose d'un grand écran, d'une souris et d'un clavier, ce qui permet d'afficher plus d'informations simultanément et d'utiliser des effets au survol (hover). Sur mobile, l'interface doit être pensée pour le tactile : boutons plus grands, navigation verticale (scroll), éléments bien espacés et textes lisibles.**

**Un site ergonomique doit donc être responsive, c'est-à-dire capable de s'ajuster automatiquement à toutes les tailles d'écran pour garantir une expérience optimale.**

## IV - Architecture technique

### Technologies

#### Front-end :

- React v22.15.0 associé à Vite v6.3.1
- Tailwind CSS v3.4.17 et Radio UI v1.1
- React Router DOM v7
- Axios 1.9.0
- Lucide React.

#### Back-end :

- Node.js 22.15.0 et Hono 4.7.8
- PostgreSQL avec Prisma 6.7.0
- JsonWebtoken 9.0.2 (JWT) et Bcrypt 6.0.0
- Zod 3.24.3.

#### Autre :

- Nodemailer
- Cloudinary
- Vitest

# Base de données

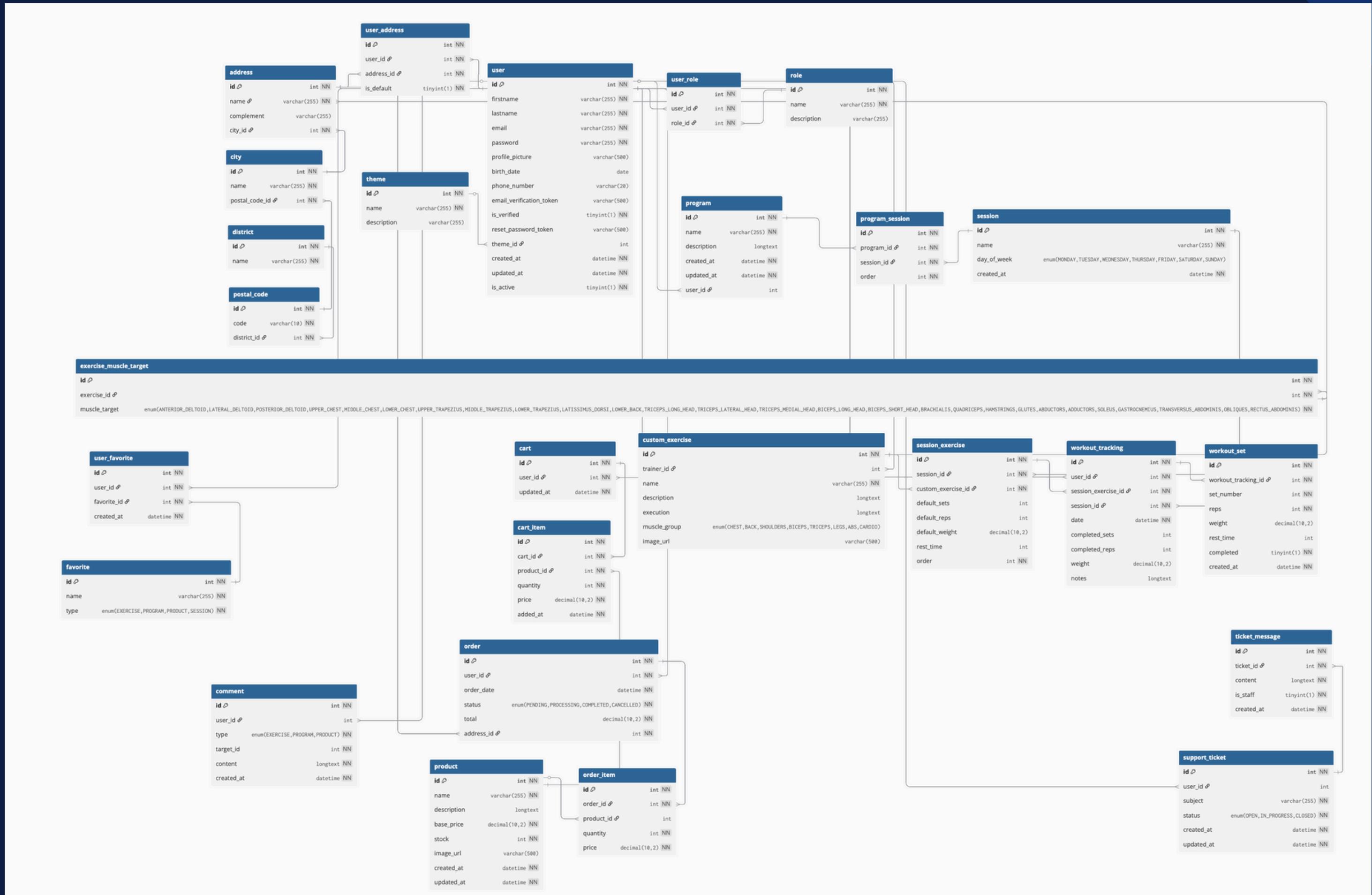
## PostgreSQL :

- Base de données relationnelle robuste
- Excellent avec Prisma
- Stable, moderne et performant
- Données structurées (utilisateurs, historiques d'entraînement.)

## Pourquoi pas de NoSQL :

- Pas de relations complexes entre les tables
- Complique la logique métier.
- Pas de garanties strictes d'intégrité
- Données de R-Training trop structurées pour NoSQL

# MLD



# Schéma Prisma

```
generator client {
  provider      = "prisma-client-js"
  output        = "../generated/prisma"
  binaryTargets = ["native", "debian-openssl-3.0.x"]
}

datasource db {
  provider = "postgresql"
  url     = env("DATABASE_URL")
}

model Role {
  id      Int      @id @default(autoincrement())
  name    String   @unique
  description String?
  users   UserRole[]
}

model User {
  id          Int      @id @default(autoincrement())
  firstname   String
  lastname    String
  email       String   @unique
  password    String
  profilePicture String?
  birthDate   DateTime?
  phoneNumber String?
  emailVerificationToken String?
  isVerified  Boolean  @default(false)
  resetPasswordToken String?
  theme       Theme?   @relation(fields: [themeId], references: [id], onDelete: SetNull)
  themeId    Int?
  createdAt   DateTime @default(now())
  updatedAt   DateTime @default(now()) @updatedAt
  isActive    Boolean  @default(true)
  role        UserRole?
  addresses   UserAddress[]
  programs    Program[]
  customExercises CustomExercise[] @relation("TrainerExercises")
  workoutTrackings WorkoutTracking[]
  userFavorites UserFavorite[]
  cart        Cart?
  orders      Order[]
  comments    Comment[]
  supportTickets SupportTicket[]
}
```

# Variables d'environnement

.env (back-end)

```
PORT=
APP_URL=
FROM_EMAIL=
SMTP_HOST=
SMTP_PORT=
SMTP_USERNAME=
SMTP_PASSWORD=
JWT_SECRET=
JWT_EXPIRES_IN=
JWT_REFRESH_EXPIRES_IN=
CLOUD_NAME=
CLOUD_API_KEY=
CLOUD_API_SECRET=
CLOUDINARY_URL=
DATABASE_URL=
```

.env (front-end)

```
VITE_API_URL=
```

# V - Développement et fonctions clés

# Connexion et inscription

```
// New Registration
async function register(data) {
  const { firstname, lastname, email, password, birthDate, phoneNumber } = data;
  console.log("Service - Données reçues:", data);

  try {
    // Check if user already exists
    const existingUser = await prisma.user.findUnique({
      where: { email }
    });

    if (existingUser) {
      throw new Error('Cette adresse email est déjà utilisée');
    }

    // Check if role already exists
    const UserRole = await prisma.role.findFirst({
      where: { name: 'user' }
    });

    if (!UserRole) {
      throw new Error('Le rôle utilisateur n\'existe pas');
    }

    // Check token
    const emailVerificationToken = generateEmailVerificationToken(email);

    // Hash password
    const hashedPassword = await hashPassword(password);

    // Create new user
    const user = await prisma.user.create({
      data: {
        firstname,
        lastname,
        email,
        password: hashedPassword,
        birthDate: birthDate ? new Date(birthDate) : undefined,
        phoneNumber,
        isVerified: false,
        emailVerificationToken
      }
    });
  }
}
```

```
// Register
authRoutes.post(
  "/register",
  zValidator('json',
    z.object({
      firstname: z.string().min(2),
      lastname: z.string().min(2),
      email: z.string().email("Adresse e-mail invalide"),
      password: z.string()
        .min(10, "Le mot de passe doit contenir au moins 10 caractères")
        .regex(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$/, "Le mot de passe doit contenir une majuscule, une minuscule et un chiffre."),
      confirmPassword: z.string(),
      birthDate: z.string()
        .regex(/^\d{4}-\d{2}-\d{2}$/, "Format de date invalide (AAAA-MM-JJ)")
        .optional(),
      phoneNumber: z.string()
        .regex(/^\(0[67] [0-9]{8}\)|\+33[67] [0-9]{8}$/, "Format de numéro invalide")
        .optional()
    }).refine((data) => data.password === data.confirmPassword, {
      message: "Les mots de passe ne correspondent pas",
      path: ["confirmPassword"]
    })
  ),
  register
);

// Login
authRoutes.post(
  "/login",
  zValidator('json',
    z.object({
      email: z.string().email("Adresse e-mail invalide"),
      password: z.string().min(10, "Le mot de passe doit contenir au moins 10 caractères")
    })
  ),
  login
);
```

# JWT et Bcrypt

## Fonction de génération de Tokens (JWT)

```
function generateToken(payload, type = 'auth') {
  if (!env.JWT_SECRET) {
    throw new Error('JWT_SECRET n\'est pas défini dans les variables d\'environnement');
  }

  const expiresIn = {
    'auth': '48h',
    'email': '24h',
    'reset': '1h',
    'refresh': '7d'
  }[type] || '1h';

  return jwt.sign(
    { ...payload, type },
    env.JWT_SECRET,
    { expiresIn }
  );
}
```

## Exemple de hachage de mot de passe (Bcrypt)

```
// Hash password
const hashedPassword = await hashPassword(password);
```

## Exemple d'expression régulière (Regex) de mots de passe (Zod)

```
password: z.string()
  .min(10, "Le mot de passe doit contenir au moins 10 caractères")
  .regex(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$/, 
        "Le mot de passe doit contenir une majuscule, une minuscule et un chiffre."),
confirmPassword: z.string(),
```

# Middleware AuthGuard

Accepte les tokens depuis Cookies HTTP-only et Headers authorization

Vérifier si un token est proche de son expiration et les renouvelle

Vérifie la structure, l'expiration et le type

Résultats :

- Validation stricte des tokens
- Flexibilité : support cookies + headers
- Rotation automatique des tokens
- Messages d'erreur claires

```
// Function to check if a token is near expiration
function isTokenNearExpiration(decodedToken, thresholdMinutes = 10) {
  if (!decodedToken.exp) return false;

  const expirationTime = decodedToken.exp * 1000; // Convert to milliseconds
  const currentTime = Date.now();
  const timeUntilExpiration = expirationTime - currentTime;

  // If token expires in less than X minutes
  return timeUntilExpiration < thresholdMinutes * 60 * 1000;
}
```

# Sécurité

## Gestion des mots de passe :

- Bcrypt + Zod + validation côté serveur et côté client

## Authentification JWT :

- Tokens JWT sécurisés avec expiration
- Différents types de tokens (auth, email, reset, refresh)
- Validation stricte des tokens avec vérification du type
- Rotation automatique des tokens proches de l'expiration
- Cookies HttpOnly pour stocker les tokens d'accès

## Protection des routes

- Middleware authGuard robuste
- Système de rôles (user, trainer, admin)
- Protection côté frontend avec RouteGuard
- Vérification des autorisations par rôle

## Upload de fichiers

- Validation des types de fichiers autorisés
- Limitation de taille (5MB)
- Vérification MIME type

# Envoi d'emails

```
let transporter;

// Create transporter for email sending only if credentials are configured
if (env.SMTP_USERNAME && env.SMTP_PASSWORD) {
  transporter = nodemailer.createTransport({
    host: env.SMTP_HOST,
    port: env.SMTP_PORT,
    secure: false, // true for 465, false for other ports
    auth: {
      user: env.SMTP_USERNAME,
      pass: env.SMTP_PASSWORD,
    },
    // Additional parameters for debugging
    logger: true,
    debug: process.env.NODE_ENV === 'development',
    // Disable certificate verification in development
    tls: {
      rejectUnauthorized: process.env.NODE_ENV === 'production'
    }
  })

  // Verify SMTP configuration at startup
  transporter.verify(function(error, success) {
    if (error) {
      console.error('SMTP configuration error:', error);
    }
  });
} else {
  console.error('SMTP configuration not found. Emails will not be sent in development mode.')
}
```

```
/** 
 * Sends an email
 * @param {string} to - Recipient email address
 * @param {string} subject - Email subject
 * @param {string} html - HTML content of the email
 * @returns {Promise<boolean>} - true if email was sent successfully
 */
export const sendEmail = async (to, subject, html) => {
  // In development, if no SMTP configuration, simulate sending
  if (!transporter) {
    return true;
  }

  const mailOptions = {
    from: `R-Training <${env.FROM_EMAIL}>`,
    to,
    subject,
    html,
    text: html.replace(/<[^>]*>/g, '') // Text version of the email
  }

  try {
    await transporter.sendMail(mailOptions)
    return true
  } catch (error) {
    console.error('Error sending email:', error)
    return false
  }
}
```

## Types d'emails envoyés

1. Email de vérification(inscription)
  2. Email de réinitialisation de mot de passe
  3. Email de renvoi de vérification
- Templates dans le fichier email.js

# Gestions des images

```
// Storage configuration
const storageConfig = {
  type: 'cloudinary', // 'local', 'cloudinary' or 'aws'
  localPath: '../front/public/images/profiles',
  allowedTypes: [
    'image/jpeg',
    'image/jpg',
    'image/png',
    'image/gif',
    'image/webp',
    'image/avif'
  ],
  maxSize: 5 * 1024 * 1024 // 5MB
};
```

```
  const imageUrl = await storageService.saveExerciseImage(exerciseData.exerciseImage, exercise.id);
  updateData.image_url = imageUrl;
} catch (error) {
  console.error('Erreur lors du traitement de l\'image d\'exercice:', error);
  throw new Error('Erreur lors du traitement de l\'image d\'exercice');
}
```

**Cloudinary :**

**Type : format d'images (JPG, PNG, GIF, WEBP, AVIF)**

**Taille maximale : 5MB**

**Fonction utilisant la config pour stocker l'image d'un exercice en cloud**

**Gère aussi la suppression d'une image pour la retirer du cloud**

# VI - Tests et déploiement

## Outils utilisés

**Render pour l'hébergement :**

- Web service pour le back-end (avec variables d'environnement)
- Static site pour le front-end (avec variables d'environnement)
- Base de données en PostgreSQL avec le Schéma Prisma

**Vitest pour les test unitaires**

# Étapes de déploiement

- **Selection du repository et de la branche en associant le compte GitHub**
- **Création de la base de données en PostgreSQL avec le schéma Prisma**
- **Configuration du Web Service (back-end) avec saisie manuelle des variables d'environnement et des commandes pour lancer le serveur et installer les dépendances.**
- **Configuration du Static Site (front-end) avec saisie manuelle des variables d'environnement et commande de lancement du service et d'installation des dépendances.**

# Gestion des .env sur Render

**Render ne permet pas d'envoyer directement un fichier .env dans le dépôt pour des raisons de sécurité.**

**À la place, toutes les variables d'environnement doivent être renseignées manuellement dans l'interface Render. Depuis le tableau de bord du service, dans les paramètres du service (onglet Environment), on peut ajouter une à une les variables nécessaires au bon fonctionnement du projet. Il est essentiel que ces variables soient bien définies avant le déploiement, sinon le serveur échouera à démarrer.**

**De plus, Render masque automatiquement les valeurs pour éviter toute fuite accidentelle. Cette approche renforce la sécurité et permet de différencier les configurations selon l'environnement (développement, production...).**

# Problèmes rencontrés

**L'installation de PostgreSQL et Prisma.**

**Problèmes de gestion de Prisma (variables).**

**Difficultés à configurer Cloudinary**

**Gestion des middleware de l'authentification et des rôlrs (permissions)**

# Tests unitaires

```
describe('Auth Controller Tests', () => {
  beforeEach(() => {
    vi.clearAllMocks();
    mockContext = { req: { valid: vi.fn() }, ... };
  });

  describe('login', () => {
    it('should successfully login and set cookies', async () => { ... });
    it('should handle unverified email with 400 status', async () => { ... });
    it('should handle invalid credentials', async () => { ... });
  });

  describe('logout', () => {
    it('should clear cookies and logout successfully', async () => { ... });
  });
});
```

```
expect(mockContext.header).toHaveBeenCalledWith(
  'Set-Cookie',
  expect.stringContaining('accessToken=mock-jwt-token')
);
expect(mockContext.json).toHaveBeenCalledWith({
  message: 'Connexion réussie',
  user: mockResult.user
});
```

```
expect(mockContext.header).toHaveBeenCalledWith(
  'Set-Cookie',
  expect.stringContaining('Max-Age=0')
);
expect(mockContext.json).toHaveBeenCalledWith({
  message: 'Déconnexion réussie'
});
```

```
expect(mockContext.json).toHaveBeenCalledWith({
  error: 'Email ou mot de passe incorrect'
}, 400);
```

```
expect(mockContext.json).toHaveBeenCalledWith(mockResult, 400);
```

```
Mac:back ryan$ npm test auth.test.js
> back@1.0.0 test
> vitest auth.test.js
```

```
DEV v3.1.2 /Applications/MAMP/htdocs/R-Training/back
✓ src/test/auth.test.js (4 tests) 3ms
  ✓ Auth Controller Tests > login > should successfully login and set cookies 2ms
  ✓ Auth Controller Tests > login > should handle unverified email with 400 status 0ms
  ✓ Auth Controller Tests > login > should handle invalid credentials 0ms
  ✓ Auth Controller Tests > logout > should clear cookies and logout successfully 0ms
Test Files 1 passed (1)
Tests 4 passed (4)
Start at 21:55:30
Duration 1.27s (transform 48ms, setup 7ms, collect 124ms, tests 3ms, environment 0ms, prepare 36ms)
```

## Test pour :

- Connexion
- Déconnexion
- Connexion échouée
- Connexion sans validation de l'email
- Résultats des tests dans le terminal

# VI - Démonstration

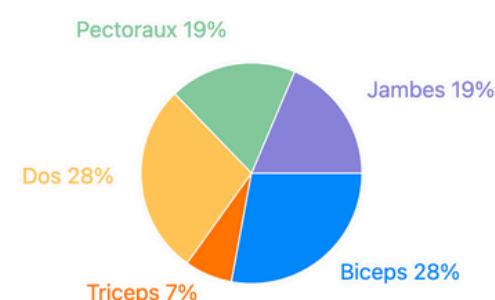
## Capture d'écran des pages

### Analyses et statistiques

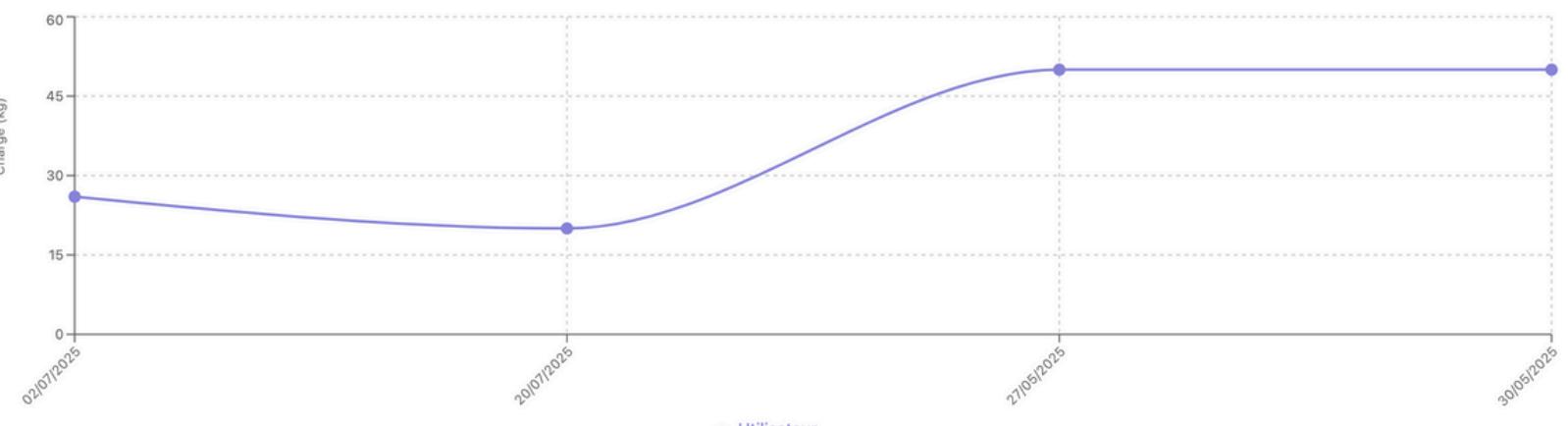
#### Répartition des entraînements par groupe musculaire

Distribution de mes séances d'entraînement par zone musculaire travaillée

Ryan Ladmia (Moi)



#### Développé incliné



#### Développé couché barre



### Mes Entraînements

Rechercher une séance, programme ou exercice... Toutes les dates

Séances enregistrées **13**

Total des exercices **43**

Total des séries **163**

dimanche 20 juillet 2025

Hypertrophie Push  
1 exercice 3 séries ~8 min

mercredi 2 juillet 2025

Force Push  
1 exercice 4 séries ~10 min

#### Exercices réalisés

##### Développé incliné CHEST

Séries **4**

Répétitions totales **24**

Poids moyen (kg) **26.0**

## Créer un nouveau programme

Créez un programme d'entraînement personnalisé avec des séances et des exercices

### Informations générales

Nom du programme \*

Ex: Programme prise de masse

Description

Description de votre programme...

### Séances d'entraînement

+ Ajouter une séance

Aucune séance ajoutée. Cliquez sur "Ajouter une séance" pour commencer.

Annuler

Créer le programme

## Ajouter un nouvel exercice

Créez un exercice personnalisé pour vos programmes d'entraînement

Nom de l'exercice \*

Ex: Développé couché

Groupe musculaire \*

Sélectionner un groupe musculaire

Description

Description de l'exercice...

Instructions d'exécution

Comment exécuter l'exercice...

Image de l'exercice



Cliquez pour choisir un fichier

## Gestion des utilisateurs

Administrez les comptes utilisateurs de la plateforme

User 2 Utilisateur 3  
user2@example.com

Utilisateur  
Actif

Trainer Coach  
trainer@example.com

Coach  
Actif

User 1 Utilisateur 1  
user@example.com

Utilisateur  
Actif

Ryan Ladmia  
ryan.ladmia@laplateforme.io

Administrateur  
Actif

Admin Admin  
admin@example.com

Administrateur  
Inactif

## VIII - Bilan personnel

### Compétences techniques acquises

PostgreSQL et Prisma

Cloudinary

Configuration de JWT Cookies HTTP-only

Hébergement sur Render

Tests unitaires

# Compétences transversales

**Autonomie sur un projet complexe**

**Structure du projet du début à la fin**

**Méthode de travail organisée et professionnelle**

**Planification des tâches en fonction des priorités**

**Organisation des fichiers**

**Gestion des erreurs**

**Sécurisation des requêtes**

**Capacité d'adaptation de d'apprentissage**

**Débogage**

# Perspectives d'évolution du projet

**Boutique en ligne : base de données déjà configurée pour cela  
Chat, FAQ, ou Blog en NoSQL**

**Système de coaching avancé : rôle trainer déjà en place  
Mise en place de notation et commentaires pour les programmes et  
exercices**

**Ajout de vidéos d'entraînement**

# Conclusion