PE69 - MathBac Boost

CAHIER DES CHARGES TECHNIQUE

Spécifications techniques et architecture

École Centrale de Lyon
Novembre 2025

1. ARCHITECTURE GÉNÉRALE

1.1 Architecture applicative

L'application suit une architecture client-serveur classique avec séparation claire entre frontend et backend. Le choix d'une webapp responsive permet une compatibilité multiplateforme sans développement natif.

Frontend (Client)

- Framework : React + Next.js
- Rendu : Server-Side Rendering (SSR) pour optimiser le SEO et les performances

Backend (Serveur)

- Langage : PHP (pour compatibilité avec hébergement économique)
- Base de données : MySQL ou PostgreSQL (selon hébergeur)
- Hébergement : VPS PulseHeberg (offre étudiante 20€/an)

Services externes

- Authentification : Firebase Auth ou Supabase Auth
- Storage : Supabase Storage (pour médias si nécessaire)

1.2 Stack technologique

Frontend

- React 18+ : bibliothèque JavaScript pour interfaces utilisateur
- Next.js 14+ : framework React avec SSR et routing
- Tailwind CSS : framework CSS utility-first pour styling rapide
- JSXGraph : bibliothèque pour graphiques mathématiques interactifs
- KaTeX : rendu rapide de formules mathématiques LaTeX

Backend

- PHP 8+ : langage serveur pour API REST
- MySQL 8+ / PostgreSQL : base de données relationnelle
- Composer : gestionnaire de dépendances PHP

DevOps & Outils

- Git & GitHub : versioning et collaboration
- npm / yarn : gestionnaire de packages JavaScript
- Trello : gestion de projet et tâches

2. MODÈLE DE DONNÉES

2.1 Entités principales

Utilisateur (User)
- id : identifiant unique (UUID)
- email : adresse email (unique)
- nom, prenom : informations personnelles
- firebase_uid : identifiant Firebase Auth
- created_at, updated_at : timestamps
- preferences : JSON (notifications, anonymat, etc.)

Chapitre (Chapter)
- id : identifiant unique
- titre : nom du chapitre
- description : résumé
- ordre : position dans le programme
- cours_content : contenu du cours (Markdown/HTML)

Exercice (Exercise)
- id : identifiant unique
- chapter_id : référence au chapitre
- type : "flash" ou "annale"
- difficulte : niveau 1-5
- enonce : texte de l'énoncé (LaTeX supporté)
- config : JSON (paramètres de génération pour exercices infinis)
- correction : JSON (étapes de correction)
- temps_estime : durée moyenne en minutes

Tentative (Attempt)
- id : identifiant unique
- user_id : référence utilisateur
- exercise_id : référence exercice
- reponse : réponse de l'élève (JSON)
- correct : booléen (réussite/échec)
- temps_passe : temps en secondes
- created_at : timestamp

Progression (Progress)
- id : identifiant unique
- user_id : référence utilisateur
- chapter_id : référence chapitre
- niveau : score 0-100
- nb_exercices_reussis : compteur
- dernier_entrainement : timestamp

Classe (Class)
- id : identifiant unique
- nom : nom de la classe
- code : code d'accès (unique)
- prof_id : référence au professeur

Duel (Duel)
- id : identifiant unique
- player1_id, player2_id : références utilisateurs
- score1, score2 : scores finaux
- status : "en_cours", "termine", "abandonne"
- created_at, finished_at : timestamps

3. API REST

3.1 Architecture API

L'API REST suit les conventions RESTful avec des endpoints organisés par ressources. Toutes les réponses sont au format JSON.

Structure des endpoints
- Base URL : https://api.mathbacboost.fr/v1
- Authentification : Bearer token (Firebase JWT)
- Format : JSON (Content-Type: application/json)

3.2 Endpoints principaux

Authentification
- POST /auth/register : création de compte
- POST /auth/login : connexion
- POST /auth/reset-password : réinitialisation mot de passe

Utilisateur
- GET /users/me : profil utilisateur
- PUT /users/me : mise à jour profil
- DELETE /users/me : suppression compte

Chapitres
- GET /chapters : liste des chapitres
- GET /chapters/:id : détail d'un chapitre
- GET /chapters/:id/cours : contenu du cours

Exercices
- GET /exercises?chapter_id=X&type=flash : liste d'exercices
- GET /exercises/:id : détail exercice
- POST /exercises/:id/generate : génération nouvelle instance
- POST /exercises/:id/submit : soumission réponse
- GET /exercises/:id/correction : récupération correction

Progression
- GET /progress : progression globale
- GET /progress/chapters : progression par chapitre
- GET /progress/stats : statistiques détaillées
- POST /progress/evaluate : lancer évaluation initiale

Duels
- POST /duels/create : créer duel
- GET /duels/:id : détails duel

- POST /duels/:id/join : rejoindre duel
- GET /duels/leaderboard : classement

Classes
- POST /classes/join : rejoindre avec code
- GET /classes/:id : détails classe
- GET /classes/:id/leaderboard : classement classe

4. MOTEUR D'EXERCICES

4.1 Génération d'exercices

Le moteur d'exercices permet la génération infinie d'exercices via paramétrage. Chaque exercice est défini par une structure JSON contenant les paramètres variables.

Structure type
- Template d'énoncé avec variables (ex: {a}, {b}, {c})
- Plages de valeurs pour chaque variable
- Formule de calcul de la réponse correcte
- Règles de validation (tolérance numérique, format)

4.2 Système de validation

Types de validation
- Numérique : comparaison avec tolérance (epsilon = 0.01)
- Algébrique : simplification et comparaison d'expressions
- Tableau de variations : vérification structure et valeurs
- Arbre de probabilités : validation des branches et probabilités
- QCM : comparaison directe

4.3 Système de feedback

- Hints progressifs (3 niveaux)
- Feedback immédiat selon type d'erreur
- Correction détaillée étape par étape
- Messages d'encouragement personnalisés

5. COMPOSANTS INTERACTIFS

5.1 Rendu mathématique

- KaTeX : rendu LaTeX côté client (formules, équations)
- Syntaxe simplifiée pour élèves (éditeur WYSIWYG)
- Support fractions, racines, puissances, intégrales, limites

5.2 Graphiques interactifs

- JSXGraph : traçage de fonctions, courbes, points
- Repère orthonormé configurable
- Zoom et déplacement

- Tangentes, dérivées, intégrales visualisables

5.3 Tableaux de variations

- Interface de construction par glisser-déposer
- Cases à remplir pour valeurs, variations, signes
- Validation automatique de la cohérence

6. ALGORITHMES PRINCIPAUX

6.1 Algorithme de recommandation

L'algorithme analyse le niveau de l'élève par chapitre et recommande des exercices adaptés pour optimiser la progression.

Critères de recommandation

- Niveau actuel de l'élève sur le chapitre (0-100)
- Taux de réussite récent
- Temps depuis dernier entraînement
- Chapitres à prioriser (lacunes identifiées)

Sélection d'exercices

- Difficulté adaptée : niveau_eleve ± 10%
- Variété des types d'exercices
- Éviter répétition d'exercices récents

6.2 Calcul du niveau

Le niveau sur chaque chapitre est calculé dynamiquement basé sur les tentatives récentes avec pondération temporelle.

Formule

- Score = (Σ réussites × difficulté × poids_temporel) / nb_tentatives
- Poids temporel : décroissance exponentielle (λ = 0.1 par jour)
- Normalisation sur échelle 0-100

7. SÉCURITÉ

7.1 Authentification

- Firebase Auth : gestion sécurisée des identifiants
- JWT tokens : expiration 1 heure, refresh token 30 jours
- Hashage bcrypt pour mots de passe (si auth custom)
- Rate limiting : 5 tentatives max / 15 min

7.2 Protection des données

- HTTPS obligatoire (TLS 1.3)
- Chiffrement données sensibles en base
- Conformité RGPD : consentement, droit à l'oubli, export données
- Anonymisation pour statistiques agrégées

7.3 Validation des entrées

- Sanitization côté serveur (protection XSS)
- Prepared statements (protection SQL injection)
- Validation schéma JSON pour réponses exercices

8. OPTIMISATIONS TECHNIQUES

8.1 Frontend

- Code splitting : chargement lazy des routes
- Compression images (WebP, lazy loading)
- Cache navigateur : service worker pour offline
- Minification JS/CSS en production

8.2 Backend

- Indexation base de données (user_id, exercise_id)
- Cache Redis : sessions, résultats fréquents
- Pagination API : 20 résultats max par requête
- Connection pooling pour base de données

9. STRATÉGIE DE TESTS

9.1 Tests unitaires

- Frontend : Jest + React Testing Library
- Backend : PHPUnit
- Couverture cible : > 70% fonctions critiques

9.2 Tests d'intégration

- Tests API : Postman / Newman
- Tests E2E : Cypress pour parcours utilisateur

9.3 Tests utilisateurs

- 3 sessions de tests en classe (février-mars 2026)
- Protocole : observation + questionnaire
- Recrutement via Lionel Guimont (élèves Terminale)

10. DÉPLOIEMENT

10.1 Environnements

- Développement : localhost
- Staging : Vercel preview branches
- Production : Vercel + VPS PulseHeberg

10.2 Pipeline CI/CD

- GitHub Actions : tests automatiques sur PR
- Déploiement automatique : merge sur main → production
- Rollback rapide : tags Git versionnés

11. MONITORING & MAINTENANCE

11.1 Logs

- Logs applicatifs : erreurs, requêtes lentes
- Niveau de log : ERROR, WARN, INFO, DEBUG

11.2 Métriques

- Temps de réponse API (cible < 200ms)
- Taux d'erreur (cible < 1%)
- Uptime (cible > 99%)