

**Deadline : 27 octobre 2025 (21 jours)**

---

## **PHASE 1 : Documentation Technique**

**Durée : 3-4 jours**

**Deadline : 10 octobre**

### **Task 1 - Architecture Système (FAIT)**

- ☒ Schéma d'architecture 3-tiers
- ☐ Correction finale : Node.js → Python + FastAPI

### **Task 2 - Base de Données (1-2 jours)**

- ☐ Identifier les entités/tables nécessaires
- ☐ Définir les relations entre tables
- ☐ Créer le diagramme ER (Entity-Relationship)
- ☐ Définir les attributs et types de données
- ☐ Documenter les clés primaires et étrangères

### **Task 3 - Diagrammes de Séquence (1 jour)**

- ☐ Cas d'usage 1 : "Utilisateur lance une séance"
- ☐ Cas d'usage 2 : "Utilisateur valide une séance complétée"
- ☐ Cas d'usage 3 : "Mini-compétition hebdomadaire (4 joueurs)"

### **Task 4 - Spécifications API (0.5 jour)**

- ☐ Lister les APIs externes (si nécessaire)
- ☐ Définir tous les endpoints internes (GET/POST/PUT/DELETE)
- ☐ Spécifier format entrée/sortie (JSON)

### **Task 5 - Stratégies SCM & QA (0.5 jour)**

- ☐ Définir stratégie Git (branches, workflow)
  - ☐ Définir stratégie de tests (unitaires, intégration)
  - ☐ Planifier le déploiement
- 

## **PHASE 2 : Apprentissage Python + FastAPI**

**Durée : 5-7 jours**

**Deadline : 17 octobre**

### **Jour 1-2 : Révision Python (fondamentaux)**

## Objectifs :

- ☐ Variables, types de données (int, str, list, dict)
- ☐ Fonctions (def, return, paramètres)
- ☐ Classes et objets (POO basique)
- ☐ **Type hints** (crucial pour FastAPI) : `def add(x: int, y: int) -> int:`
- ☐ List comprehensions : `[x*2 for x in range(10)]`
- ☐ Gestion d'erreurs : `try/except/finally`
- ☐ Modules et imports

## Ressources :

- Documentation Python officielle
  - Real Python (tutoriels)
  - Exercices sur HackerRank/LeetCode
- 

## Jour 3-4 : FastAPI Fondamentaux

### Objectifs :

- ☐ Installation : `pip install fastapi uvicorn`
- ☐ Créer première route GET
- ☐ Routes POST, PUT, DELETE
- ☐ Path parameters : `/users/{user_id}`
- ☐ Query parameters : `/items?skip=0&limit=10`
- ☐ Request Body avec Pydantic
- ☐ Response models
- ☐ Documentation auto à `/docs`

### Projet pratique : Créer une mini-API "Todo List"

- ☐ GET /todos (lister toutes les tâches)
- ☐ POST /todos (créer une tâche)
- ☐ PUT /todos/{id} (modifier une tâche)
- ☐ DELETE /todos/{id} (supprimer une tâche)

### Ressources :

- <https://fastapi.tiangolo.com/tutorial/>
  - YouTube : "FastAPI en 2h" (Docstring)
- 

## Jour 5-6 : FastAPI Avancé + Base de Données

### Objectifs :

- ☐ SQLAlchemy : connexion PostgreSQL
- ☐ Définir models SQLAlchemy
- ☐ CRUD operations avec SQLAlchemy
- ☐ Authentification JWT
- ☐ Middleware CORS (pour le front-end)
- ☐ Variables d'environnement (.env)
- ☐ Structure de projet propre

### Structure recommandée :

```
fitness-clash-backend/  
|  
├─ app/  
|   ├── __init__.py  
|   ├── main.py          # Point d'entrée FastAPI  
|   ├── database.py      # Connexion BDD  
|   ├── models.py        # Models SQLAlchemy  
|   ├── schemas.py       # Pydantic schemas  
|   ├── crud.py          # Opérations CRUD  
|   └─ routers/  
|       ├── users.py  
|       ├── workouts.py  
|       └─ competitions.py  
|  
├─ tests/  
|   └─ test_main.py  
|  
├─ requirements.txt  
├─ .env  
└─ README.md
```

### Ressources :

- FastAPI docs : SQL Databases
- Tutorial SQLAlchemy + FastAPI

---

## ✅ Jour 7 : Tests & Consolidation

### Objectifs :

- ☐ Tester tous les concepts appris
- ☐ Améliorer la mini-API Todo List
- ☐ Ajouter authentification JWT à la Todo List
- ☐ Résoudre les incompréhensions
- ☐ Préparer questions techniques

# PHASE 3 : Développement FITNESS CLASH

**Durée : 10-12 jours**

**Deadline : 27 octobre**

## **Sprint 1 : Base de l'API (3-4 jours)**

**Deadline : 20 octobre**

### **Jour 1 : Setup du projet**

- ☐ Créer repo GitHub
- ☐ Structure de projet (voir ci-dessus)
- ☐ Setup PostgreSQL (local ou cloud)
- ☐ Connexion BDD testée
- ☐ Premier commit

### **Jour 2-3 : Modèles de données**

- ☐ Models SQLAlchemy :
- ☐ User (id, username, email, password\_hash, created\_at)
- ☐ Exercise (id, name, category, difficulty, description, animation\_url)
- ☐ Workout (id, user\_id, date, completed, score)
- ☐ WorkoutExercise (workout\_id, exercise\_id, reps, duration)
- ☐ Competition (id, week, participants[], winner\_id, date)
- ☐ Score (id, user\_id, competition\_id, time, rank)

### **Jour 4 : CRUD basique**

- ☐ Routes Users : POST /register, POST /login
- ☐ Routes Exercises : GET /exercises, GET /exercises/{id}
- ☐ Routes Workouts : GET /workouts, POST /workouts
- ☐ Tests manuels via </docs>

---

## **Sprint 2 : Logique Métier (3-4 jours)**

**Deadline : 23 octobre**

### **Authentification**

- ☐ Hashing de mots de passe (bcrypt)
- ☐ Génération JWT tokens
- ☐ Middleware de vérification token
- ☐ Protection des routes privées

### **Génération de séances**

- ☐ Algorithme de génération aléatoire :
- ☐ 1 échauffement
- ☐ 3 exercices (haut/tronc/bas du corps variés)
- ☐ 1 étirement
- ☐ Route : POST /workouts/generate
- ☐ Respect des contraintes (difficulté, catégories)

### Validation de séances

- ☐ Route : POST /workouts/{id}/complete
- ☐ Calcul du score (temps, difficulté)
- ☐ Historique utilisateur

### Alternative d'exercices

- ☐ Route : POST /workouts/{workout\_id}/skip-exercise/{exercise\_id}
- ☐ Suggestion d'alternative (même catégorie, difficulté -1)



## Sprint 3 : Mini-Compétition (2-3 jours)

**Deadline : 25 octobre**

### Logique de compétition

- ☐ Route : POST /competitions/create
- ☐ Système de matching (4 joueurs)
- ☐ Génération séance de compétition (basée sur trainings de la semaine)
- ☐ Système bonus/malus aléatoire
- ☐ Calcul classement final

### Endpoints compétition

- ☐ GET /competitions/weekly (compétition en cours)
- ☐ POST /competitions/{id}/join (rejoindre compétition)
- ☐ POST /competitions/{id}/complete (soumettre résultat)
- ☐ GET /competitions/{id}/leaderboard (classement)



## Sprint 4 : Polish & Tests (2 jours)

**Deadline : 27 octobre**

### Tests

- ☐ Tests endpoints critiques (pytest)
- ☐ Tests logique métier
- ☐ Tests d'intégration

## Corrections & Amélioration

- ☐ Gestion d'erreurs propre (messages clairs)
- ☐ Validation des inputs renforcée
- ☐ Logging des erreurs
- ☐ Code review entre équipiers

## Documentation

- ☐ README.md complet (installation, usage)
- ☐ Swagger docs à jour
- ☐ Commentaires dans le code






## Déploiement

- ☐ Déploiement sur Render / Railway
- ☐ Variables d'environnement configurées
- ☐ Tests en production
- ☐ URL publique fonctionnelle

---






## POINTS D'ATTENTION CRITIQUES

### NE PAS FAIRE

-  **Commencer à coder avant d'avoir fini la doc technique**
  - Résultat : Tu vas refaire 10 fois, perte de temps énorme
-  **Coder FITNESS CLASH sans avoir pratiqué FastAPI avant**
  - Résultat : Tu vas galérer sur des concepts de base
-  **Travailler en solo sans communiquer avec l'équipe**
  - Résultat : Code incompatible, conflits Git, stress
-  **Coder sans tests**
  - Résultat : Bugs découverts au dernier moment
-  **Attendre la dernière semaine pour déployer**
  - Résultat : Problèmes de déploiement non anticipés

---

### BONNES PRATIQUES

1.  **Commits réguliers sur Git**
    - Minimum 1 commit par jour
    - Messages clairs : "feat: add user authentication", "fix: workout generation bug"
  2.  **Code reviews en équipe**
    - Chaque merge request doit être reviewée
    - Améliore la qualité et la compréhension collective
  3.  **Documentation au fur et à mesure**
    - Ne pas attendre la fin pour documenter
    - README à jour en permanence
  4.  **Communication quotidienne**
    - Daily standup (même 10 min)
    - Qui fait quoi ? Blocages ? Aide nécessaire ?
  5.  **Tester au fur et à mesure**
    - Après chaque feature : test manuel + automatique
    - Ne pas accumuler la dette technique
- 



## RÉPARTITION DU TRAVAIL (Équipe de 4)

### Personne 1 : Back-End Lead (TOI)

- ☐ Architecture API
- ☐ Authentification
- ☐ Logique métier (génération séances, compétitions)
- ☐ Base de données

### Personne 2 : Front-End Lead

- ☐ Interface React
- ☐ Appels API
- ☐ Gestion d'état (Redux/Context)
- ☐ Routing

### Personne 3 : UI/UX & Animations

- ☐ Design de l'interface
- ☐ Animations stickman
- ☐ CSS/Tailwind
- ☐ Responsive design

### Personne 4 : DevOps & QA

- ☐ Setup CI/CD
- ☐ Tests automatisés
- ☐ Déploiement
- ☐ Monitoring

**Note :** Tout le monde doit pouvoir expliquer l'ensemble du projet au Demo Day !

---



## RESSOURCES ESSENTIELLES

### Python

- [Python Documentation](#)
- [Real Python Tutorials](#)

### FastAPI

- [FastAPI Documentation](#)
- [FastAPI Tutorial \(vidéo FR\)](#)

### SQLAlchemy

- [SQLAlchemy ORM Tutorial](#)

### PostgreSQL

- [PostgreSQL Tutorial](#)

### Git

- [Git Guide](#)

### Déploiement

- [Render.com](#) (gratuit)
  - [Railway.app](#) (gratuit)
- 



## CHECKLIST DEMO DAY

### Documentation

- ☐ Schéma d'architecture finalisé
- ☐ Diagramme ER de la base de données
- ☐ Diagrammes de séquence (3 cas d'usage)
- ☐ Spécifications API complètes
- ☐ README.md professionnel

### Technique



- ☐ API déployée et accessible
- ☐ Documentation Swagger fonctionnelle (</docs>)
- ☐ Front-end déployé et connecté
- ☐ Base de données en production
- ☐ Tous les endpoints testés

## Présentation

- ☐ Slides de présentation (10-15 minutes)
  - ☐ Démo live préparée (5 minutes)
  - ☐ Backup plan si problème technique
  - ☐ Chacun connaît sa partie à présenter
- 

## 💪 MESSAGE DE MOTIVATION

**Tu as 21 jours pour réaliser un projet pro.**

C'est **faisable** si tu :

1. ☒ Suis ce planning
2. ☒ Restes focus
3. ☒ Communiques avec ton équipe
4. ☒ N'abandonnes pas face aux bugs (c'est normal !)
5. ☒ Demandes de l'aide quand tu bloques

**Chaque bug résolu = une compétence gagnée.**

**Le Demo Day n'est pas juste une présentation, c'est la preuve que tu peux mener un projet de A à Z.**

**Let's go champion ! 🚀🔥**

---

## 📞 SUPPORT

Si tu bloques sur un concept ou un bug :

- 🤖 Utilise Claude (moi) pour les explications détaillées
- 🤖 Utilise GPT pour les longues conversations
- 👥 Demande à tes camarades Holberton
- 📖 Consulte la documentation officielle
- 💬 Cherche sur Stack Overflow

**Ne reste JAMAIS bloqué plus de 2h sur un problème sans demander de l'aide !**

