Antony Lignac, Baptiste Ballesteros, Alexandre Pumpalovic, Lilian Hammache

**Objectif du projet**

Ce projet est un site web de jeu de blackjack développé en Svelte pour le front-end et Symfony pour le back-end. Il permet à des utilisateurs de s’inscrire, se connecter, et jouer à des parties de blackjack, tout en conservant un historique de leurs parties et en pouvant supprimer les parties.

# Backend - API REST Symfony

### **Langage et Framework**

* **Langage** : PHP
* **Framework** : Symfony 7.x
* **Type** : API REST

### **Dépendances Principales**

* **Authentification** : LexikJWTAuthenticationBundle (JWT)
* **ORM** : Doctrine ORM
* **Sérialisation** : Symfony Serializer

### **Fonctionnalités Backend**

* Gestion des utilisateurs (inscription, connexion)
* Authentification JWT
* Logique métier du jeu de blackjack
* API REST pour les actions de jeu
* Gestion des sessions de jeu

## **Caractéristiques Techniques**

### **Performance**

* **Utilisateurs simultanés estimés** : ~100 utilisateurs journaliers
* **Architecture** : Microservices découplés
* **Communication** : API REST synchrone

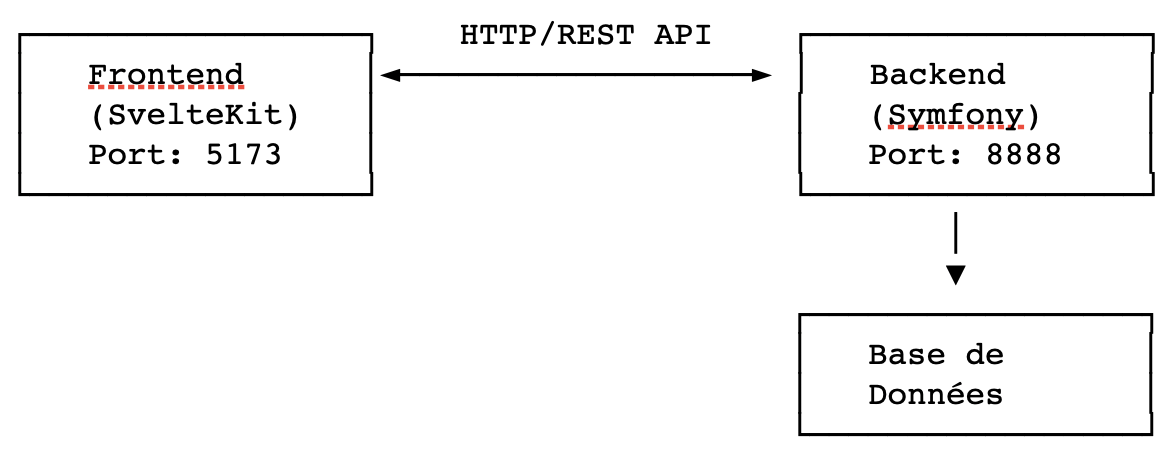
### **Environnement de Développement**

* **Conteneurisation** : Développement en containers Docker
* **Hot Reload** : Support des modifications en temps réel
* **Isolation** : Services isolés par containers

### **Outils de Développement**

* **Gestionnaire de paquets** : Composer
* **CLI** : Symfony CLI
* **ORM** : Doctrine avec migrations
* **Serveur de développement** : Symfony Local Server

## **Architecture Générale**



## 

# Justification de l'utilisation de Postman pour les tests API

## **Pourquoi Postman pour tester l'API Blackjack ?**

### Contexte du Projet

L'application Blackjack utilise une **architecture découplée** avec un backend Symfony 7 exposant une API REST pure (sans interface graphique intégrée) et un frontend SvelteKit séparé. Cette séparation nécessite des outils spécialisés pour tester efficacement chaque couche indépendamment.

## **Justifications Techniques et Méthodologiques**

### Organisation Structurée des Tests

**Avantages** :

* **Hiérarchisation claire** des endpoints par domaine métier
* **Réutilisation** des configurations d'authentification
* **Maintenance facilitée** des tests par module

### Interface Graphique pour API Backend Pure

Le backend Symfony étant une **API REST sans interface web**, Postman fournit :

* **Visualisation intuitive** des réponses JSON
* **Formatage automatique** des données de retour
* **Interface graphique** pour une API qui n'en possède pas
* **Feedback visuel immédiat** sur les codes de statut HTTP

### Collaboration et Partage d'Équipe

**Partage Rapide et Structuré**

* **Export/Import** de collections complètes
* **Documentation automatique** générée à partir des requêtes
* **Synchronisation cloud** pour l'équipe QA
* **Versionning** des collections de tests

**Standardisation des Tests**

* **Templates de requêtes** réutilisables
* **Variables d'environnement** partagées (DEV, STAGING, PROD)
* **Scripts de pré/post-requêtes** pour l'automatisation

### Gestion Avancée de l'Authentification JWT

**Fonctionnalités** :

* **Gestion automatique** des tokens JWT
* **Refresh automatique** des tokens expirés
* **Variables d'environnement** pour les tokens
* **Héritage** de l'authentification dans toute la collection

### Tests de Séquences de Jeu

1. **Authentication** → Token JWT récupéré
2. **Création de partie** → Game ID généré
3. **Actions de jeu** → Hit/Stand avec validation
4. **Fin de partie** → Résultat et statistiques
5. **Nettoyage** → Suppression des données de test

L'un des principaux avantages de Postman réside dans sa capacité à **automatiser l'enchaînement** de ces étapes critiques. Grâce aux scripts de pré/post-requêtes et à la gestion des variables d'environnement, chaque étape alimente automatiquement la suivante

# **Le lien postman**

https://github.com/LignacAntony/blackjack/blob/master/documents/BLACKJACK.postman\_collection.json

**Application Front - Svelte**

Dans un premier temps pour tester le front nous allons faire des test manuels sur l’inscription, la navigation, la connexion et la création de partie, la suppression de partie et le logout.   
En effet, faire des tests manuels sur l'inscription permet de vérifier la validité des champs, les messages d'erreur, les redirections et de tester facilement différents cas de saisie. Sur la navigation, cela permet de s'assurer que les liens fonctionnent, que les pages sont accessibles selon la connexion, et de repérer des problèmes visuels ou de logique. Pour la connexion, on vérifie la validation des identifiants, les erreurs en cas d’échec, la création de session et le retour utilisateur. Pour la création de partie, cela permet de tester l’affichage du jeu, les interactions et le déroulement complet d’une partie, la suppression d’un partie et la déconnexion.   
Globalement, nous utilisons les tests manuels pour détecter rapidement des bugs, valider l’expérience utilisateur et préparer les tests automatisés.

Lien vers test manuel : <https://docs.google.com/spreadsheets/d/1lt15GfyKNpF835_LqqwMj9XIIe0amzMdKeYEQs3AL8U/edit?usp=sharing>  
ou  
https://github.com/LignacAntony/blackjack/blob/master/documents/Test%20\_%20Test%20manuel.xlsx