Juventus Prono Bot — Discord

Procédure — Bot Discord de pronostics Juventus FC

Développeur	COUR Cyrille
Type de projet	Projet personnel
Stack	Python 3.13+ / discord.py / SQLite3
Date	24/08/2026
Version	1.0

Sommaire

1. Contexte

Juventus Prono Bot est un projet personnel développé par envie de suivre les matchs de la Juventus FC, club de cœur de l'auteur, et d'organiser des pronostics entre amis directement sur Discord. Le bot automatise la publication de sondages avant chaque match, le suivi de l'état des rencontres et l'annonce des résultats avec les buteurs.

Note — Ce document a pour but de garder une trace claire de l'architecture, du fonctionnement et des points d'attention du bot, dans un but personnel de suivi et de maintenance future.

2. Environnement

2.1 Prérequis

- Python 3.13 ou supérieur
- Bibliothèque discord.py
- Bibliothèque python-dotenv (gestion des variables d'environnement)
- Un compte et une application Discord (bot token + permissions)
- Environnement de développement Laragon (Windows)
- Accès aux endpoints REST de l'API ESPN (Schedule & Summary)

2.2 Installation

```
$ git clone <repo-juventus-prono-bot>
$ cd Juventus-Prono-BOT
$ pip install -U discord.py python-dotenv
$ copy .env.example .env   # puis renseigner DISCORD_TOKEN
$ python main.py
```

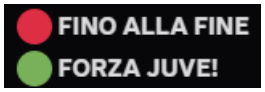
Attention — Le fichier .env n'est jamais versionné (à ajouter au .gitignore) : il contient le token du bot Discord.

3. Procédure

3.1 Cycle de vie du bot

Détail technique

Au démarrage, le bot se connecte à Discord et publie automatiquement le message "⚪🟢 FORZA JUVE!" dans le salon dédié. À l'arrêt, la fermeture du bot est interceptée pour envoyer "🔴🟢 FINO ALLA FINE" avant déconnexion.

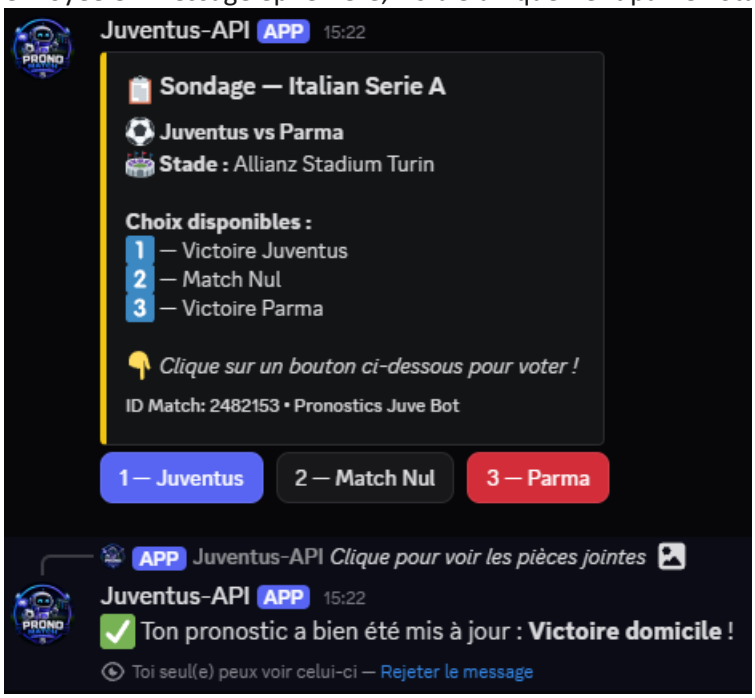


3.2 Publication des sondages de pronostics

Détail technique

Une tâche d'arrière-plan interroge périodiquement l'API ESPN afin de détecter le prochain match de la Juventus. 24 heures avant le coup d'envoi, le bot publie automatiquement un sondage dans le salon dédié, sous la forme de boutons interactifs (discord.ui.View) permettant de voter en un clic : Victoire Domicile, Match Nul, Victoire Extérieur.

Chaque vote est enregistré en base SQLite via une requête INSERT OR REPLACE INTO, ce qui permet à un utilisateur de changer d'avis tant que le match n'a pas commencé. Une confirmation de vote est envoyée en message éphémère, visible uniquement par le votant.



3.3 Suivi et clôture automatique des matchs

Détail technique

Une boucle planifiée (tasks.loop) vérifie toutes les 30 minutes l'état des matchs marqués comme OPEN en base de données. Lorsqu'un match est terminé, le bot publie une annonce de fin de match avec le score final et identifie automatiquement les utilisateurs gagnants en comparant leur vote au résultat réel.

Les buteurs et leurs minutes de jeu sont extraits dynamiquement depuis la clé details (scoringPlay) de la réponse de l'API ESPN (endpoint Summary).

3.4 Commandes Discord disponibles

Détail technique

Commande	Description
!match	Force la publication manuelle du sondage pour le prochain match
!resultat_dernier	Affiche le score, la compétition et les buteurs du dernier match joué
!force_recap	Simule l'annonce de fin de match et liste les gagnants enregistrés en BDD

4. Structure / Données

La base de données SQLite (prono_bot.db) est générée automatiquement au premier lancement et repose sur deux tables principales.

4.1 Table matches

Colonne	Type	Description
event_id	TEXT (PK)	Identifiant unique du match (ESPN)
home_team	TEXT	Nom de l'équipe à domicile
away_team	TEXT	Nom de l'équipe à l'extérieur
message_id	INTEGER	ID du message Discord contenant le sondage
status	TEXT	Statut du match (OPEN ou CLOSED)

4.2 Table votes

Colonne	Type	Description
user_id	INTEGER	ID Discord du votant
event_id	TEXT	ID du match concerné
choice	INTEGER	1 = Domicile, 2 = Nul, 3 = Extérieur

Note — Contrainte : clé primaire composée (user_id, event_id) sur la table votes, ce qui garantit un seul vote actif par utilisateur et par match.

4.3 Architecture des fichiers

```
Juventus-Prono-BOT/
├── .env                  # Variables d'environnement (DISCORD_TOKEN)
├── database.py           # Fonctions CRUD SQLite3
├── main.py               # Logique du bot, tâches loop et commandes
└── prono_bot.db          # Fichier BDD SQLite (généré auto)
```

5. Déploiement / CI-CD

Le bot est destiné à un usage personnel et tourne en environnement local (Laragon / Windows). Aucune pipeline CI/CD n'est en place à ce stade : le déploiement consiste à lancer le script main.py sur la machine hôte, avec le fichier .env correctement renseigné.

- Secret requis : DISCORD_TOKEN (token du bot, à générer depuis le portail développeur Discord)
- Le bot doit disposer des intents nécessaires (message content, etc.) activés dans le portail développeur
- Aucune base de données externe : SQLite embarqué, fichier prono_bot.db généré automatiquement

6. Points de vigilance

Attention — Le token Discord ne doit jamais être commité dans le dépôt Git ; vérifier systématiquement la présence de .env dans le .gitignore.

Attention — La disponibilité des données (calendrier, scores, buteurs) dépend entièrement de l'API publique ESPN, qui n'est pas garantie contractuellement et peut changer sans préavis.

Note — La boucle de vérification toutes les 30 minutes implique un délai maximal équivalent avant la clôture effective d'un match terminé et l'annonce des résultats.