ChatGPT

Plan de Projet : Robot de Trading Algorithmique Avancé

Introduction et Objectifs du Projet

Ce projet vise à développer un **robot de trading avancé** capable de fonctionner à la fois en entraînement sur données historiques et en exécution en temps réel. L'objectif est de combiner plusieurs composantes de pointe : (1) des **modèles quantitatifs et d'apprentissage automatique** pour estimer la juste valeur ou la direction future d'un actif (« pricing »), (2) **plusieurs stratégies de trading complémentaires** (par ex. momentum, arbitrage statistique, stratégies *event-driven* basées sur les nouvelles, etc.), et (3) un module d'**analyse de nouvelles financières par LLM** (modèle de langage de type GPT) pour évaluer en temps réel le sentiment ou l'impact des actualités sur les marchés. L'intégration de **plusieurs stratégies** est essentielle, car s'appuyer sur une seule stratégie, même robuste, peut exposer à des risques importants ; une approche multi-stratégie peut améliorer les rendements ajustés du risque en diversifiant les sources de profit [1]. Le résultat attendu est un système cohérent, **entièrement automatisé**, capable de *backtester* sur le passé puis de s'adapter aux flux de données en direct, le tout selon une architecture modulaire claire et présentable en entretien.

Choix de la Classe d'Actifs Cible

Étant donné que l'utilisateur n'a pas fixé d'actif spécifique, deux classes d'actifs sont proposées pour ce projet, chacune présentant des atouts particuliers :

- **Option 1 : Actions (marchés boursiers)** – Les actions (par exemple les grandes capitalisations ou indices boursiers) constituent un univers familier et riche en données. Elles offrent un large **historique de prix** fiable, des volumes conséquents et de nombreuses sources d'information (rapports de résultats, actualités d'entreprises, indicateurs macroéconomiques, etc.). Ces données abondantes facilitent l'entraînement de modèles d'**estimation de prix**. De plus, les marchés actions sont propices à une gamme variée de stratégies : on y observe des effets de **momentum** (poursuite de tendance) bien documentés, des opportunités d'**arbitrage statistique** (ex. paires d'actions corrélées) et des stratégies **event-driven** autour des événements d'entreprise (fusions-acquisitions, annonces de bénéfices, etc.). L'analyse de nouvelles par LLM est particulièrement pertinente sur les actions, car les nouvelles spécifiques aux entreprises ou secteurs ont souvent un impact directionnel clair (positif/négatif) sur les cours.

- **Option 2 : Cryptomonnaies** – Les cryptomonnaies (par ex. Bitcoin, Ethereum) offrent un terrain d'expérimentation moderne, disponible 24h/24, avec une **volatilité élevée**. Cette volatilité en fait des actifs idéaux pour les stratégies de **momentum**, où les traders cherchent à capter les mouvements rapides des prix [2]. Le marché crypto est fragmenté en de multiples plateformes, ce qui crée des **opportunités d'arbitrage** (écarts de prix entre exchanges) exploitables par un robot réactif. Les crypto-actifs sont également très influencés par les nouvelles (réglementations, tweets de personnalités, évolutions techniques des protocoles). Un LLM pourrait analyser en temps réel des flux d'actualités spécialisées (par ex. annonces de régulateurs, actualités technologiques blockchain) et en déduire un **sentiment de marché**. Enfin, les données

historiques crypto sont facilement accessibles via des API publiques, et la communauté offre de nombreux exemples de stratégies quantitatives, ce qui peut nourrir la conception du modèle.

(Selon l'intérêt du public cible de l'entretien, on peut choisir l'option actions pour un contexte plus traditionnel, ou l'option cryptomonnaies pour un projet plus innovant. Dans les deux cas, le cadre multi-stratégie et l'utilisation d'un LLM restent pertinents.)

Sources de Données et Pipeline de Données

Un **pipeline de données robuste** est crucial pour alimenter le robot, à la fois lors de la phase d'entraînement sur historique et en mode temps réel. Voici l'architecture proposée pour la gestion des données :

- **Données Historiques** : Une base de données ou un data lake contiendra l'historique des prix de l'actif choisi (par ex. plusieurs années de cours quotidiens ou intra-journaliers pour les actions, ou historiques minute par minute pour les cryptos), ainsi que les archives des actualités pertinentes (flux de news financières horodatées, archives de réseaux sociaux financiers, etc.). Ces données historiques permettront d'entraîner les modèles de pricing (par apprentissage supervisé sur les tendances passées) et de **backtester** chaque stratégie. Par exemple, on collectera les prix via des API (Yahoo Finance, Bloomberg aux format CSV, API d'exchange crypto, etc.) et les nouvelles via des agrégateurs (RSS feeds, APIs comme NewsAPI ou des bases de données d'actualités). Les données seront nettoyées et stockées (dans une base SQL ou une solution de stockage cloud) pour un accès efficace lors de l'entraînement et des tests.

- **Données en Temps Réel** : En mode exécution live, le robot consommera deux flux de données principaux. D'une part, un **flux de marché en continu** pour les prix et volumes de l'actif (par ex. via une connexion WebSocket à un fournisseur comme Finnhub ou Binance). Ce flux fournira les dernières cotations, généralement en temps quasi-réel (échelle de la seconde ou de la milliseconde si nécessaire). *Exemple*: pour des actions, on peut utiliser l'API de streaming de prix de Finnhub – les cours et volumes seraient ingérés continuellement via WebSocket et mis en buffer pour analyse [3]. D'autre part, le robot recevra un **flux de nouvelles en temps réel** : par exemple, en interrogeant périodiquement une API d'actualités financières (telle que NewsAPI) pour les derniers titres concernant l'actif ou le marché, ou en écoutant des flux spécifiques (tweets de personnalités clés, posts Reddit/Discord pour les cryptos, communiqués de presse, etc.). Ces textes bruts seront transmis au module LLM pour analyse (voir section dédiée).

- **Pipeline d'Intégration** : Les données temps réel seront acheminées au **moteur de stratégie** via un pipeline en streaming. Concrètement, on peut concevoir un composant de **Data Ingestion** qui reçoit en parallèle les tickers de marché et les nouvelles, les aligne sur une **même timeline** (par exemple, en estampillant chaque nouvelle avec l'horloge du marché) puis les diffuse aux sous-systèmes concernés. Une approche possible est l'utilisation d'une architecture événementielle (par ex. des messages dans une file ou un bus d'événements type Kafka) pour que chaque nouvelle donnée (tick de prix ou nouvelle importante) déclenche le recalcul des indicateurs et signaux de trading. En complément, un **stockage temps réel** (en mémoire ou en base temporaires) conservera un historique glissant des dernières données afin de permettre aux modèles (techniques ou ML) d'avoir du contexte (par ex. fenêtre de n minutes de prix pour calculer un indicateur technique).

Ce pipeline garantit que le modèle peut facilement passer du mode historique (lecture en batch de la base de données) au mode temps réel (lecture en continu des flux), en réutilisant les mêmes étapes de

transformation des données. La **qualité et latence** des données sont cruciales : il faudra s'assurer que le flux temps réel est suffisamment rapide (quelques centaines de millisecondes maximum de retard) et que les données sont vérifiées (par ex. filtrer les anomalies, données manquantes ou duplications) avant d'être exploitées par les stratégies.

Techniques d'Estimation et de « Pricing » de l'Actif

Pour estimer la valeur de l'actif ou prédire son évolution, le robot intégrera plusieurs **approches de modélisation** qui pourront fonctionner en parallèle :

- **Modèles quantitatifs traditionnels** : On utilisera des modèles éprouvés de valorisation ou de prévision. Par exemple, pour des actions, un **modèle factoriel** (inspiré des Fama-French, ou d'un modèle de score fondamental) pourrait estimer un prix théorique ou un rendement attendu en fonction de multiples indicateurs (valorisation, croissance des bénéfices, taux d'intérêt, etc.). Si l'on traite des dérivés (options), des modèles comme **Black-Scholes** ou des arbres binomiaux peuvent fournir le prix théorique de l'option par rapport auquel détecter des écarts. Pour les séries temporelles financières classiques, on pourra déployer un modèle ARIMA (AutoRegressive Integrated Moving Average) pour prévoir les fluctuations à court terme ; toutefois, ces modèles statistiques linéaires fonctionnent bien sur des données séquentielles régulières et des tendances saisonnières, mais **ne capturent pas les comportements non linéaires** et ne sont pas fiables sur de longues horizons ou des marchés très changeants [4]. En résumé, les modèles quantitatifs offrent une base interprétable et rapide, mais peuvent manquer de puissance prédictive dans des marchés complexes.

- **Modèles d'apprentissage automatique (ML)** : En parallèle, le robot s'appuiera sur des modèles ML entraînés sur les données historiques pour **prédire les mouvements de prix** ou la probabilité d'un scénario haussier/baissier. Plusieurs algorithmes pourront être testés et combinés en ensemble (*ensemble learning*) afin de tirer parti de leurs forces respectives [5]. Par exemple, une **forêt aléatoire (Random Forest)** peut être utilisée pour prédire la variation journalière d'un actif en se basant sur de multiples caractéristiques (techniques et fondamentales) ; ce type de modèle est réputé pour sa précision sur de grands jeux de données et sa capacité à modéliser des relations non linéaires entre variables [6]. De même, des modèles comme les **réseaux de neurones profonds** (y compris des LSTM ou des transformers adaptés aux séries temporelles financières) peuvent détecter des patterns complexes dans l'historique des prix. L'entraînement de ces modèles se fera sur le jeu de données historiques, en incluant possiblement des **features** additionnelles comme des indicateurs techniques (moyennes mobiles, RSI, etc.) ou des mesures de sentiment agrégées des nouvelles. On veillera à valider ces modèles hors-échantillon et à éviter le surapprentissage (*overfitting*). Enfin, l'approche ML pourra inclure un mécanisme d'**apprentissage en ligne** (online learning) : par exemple, une mise à jour périodique du modèle avec les nouvelles données de marché pour s'adapter aux changements de régime (voire l'utilisation de techniques adaptatives ou de renforcement si pertinent).

- **Combinaison des approches** : Le système de pricing pourrait in fine agréger les prédictions de plusieurs modèles pour plus de robustesse. Par exemple, si le modèle factoriel indique que l'actif est sous-évalué par rapport à ses fondamentaux et que simultanément le modèle ML prédit un momentum haussier, le signal d'achat est renforcé. Au contraire, si les modèles divergent, le système pourrait réduire la confiance du signal ou s'abstenir. Cette approche multicritère permet de couvrir différents aspects du prix : la valeur « juste » à long terme d'une part, et la dynamique de court terme d'autre part.

Stratégies de Trading Avancées et Complémentaires

Plusieurs stratégies algorithmique seront implémentées au sein du **moteur de trading**, afin de tirer parti de différentes situations de marché. Les stratégies sélectionnées sont **complémentaires** par leurs logiques (certaines suivent la tendance, d'autres la contrarient, certaines réagissent aux nouvelles, etc.), ce qui aide à diversifier le portefeuille du robot. Voici les principales stratégies prévues :

- **Stratégie Momentum (suivi de tendance)** – Cette stratégie cherche à profiter de la persistance des tendances des prix. L'idée est d'**acheter un actif dont le prix monte** (ou vendre un actif dont le prix baisse), en anticipant que la tendance se prolonge. Concrètement, le robot calculera des signaux de momentum via des indicateurs techniques tels que des moyennes mobiles (ex. un croisement de moyenne mobile courte et longue), l'indicateur de force relative (RSI) ou la détection de *breakouts*. Un modèle de machine learning peut également renforcer ces signaux en prédisant la probabilité que la tendance continue sur la base de configurations historiques similaires. La stratégie momentum fonctionne particulièrement bien dans des marchés **trending** et volatils – par exemple, les marchés de cryptomonnaies, très volatils, se prêtent à ce type d'approche où suivre la tendance dominante peut s'avérer payant [2] . On note cependant que le momentum peut échouer dans des marchés sans direction ou subissant des retournements brutaux ; il est donc crucial d'appliquer des **règles de gestion du risque** (stop-loss, take-profit) et de surveiller la qualité de la tendance. *À noter:* La littérature et l'expérience montrent que les stratégies de momentum peuvent être profitables dans de nombreux contextes, mais qu'elles demandent une surveillance fine : elles sont **dépendantes du régime de marché** et exigent une expertise pour gérer les phases volatiles et éviter les faux signaux [7] . L'avantage d'automatiser cette stratégie est de pouvoir analyser en continu les indicateurs et d'agir à la milliseconde près dès qu'un signal solide est identifié, sans la moindre hésitation psychologique.

- **Stratégie d'Arbitrage Statistique** – L'arbitrage vise à exploiter des **déséquilibres de prix temporaires** entre actifs fortement liés. Typiquement, il s'agit d'identifier deux (ou plusieurs) instruments dont les prix présentent une relation historique stable (par exemple deux actions du même secteur, ou le prix d'un ETF par rapport à ses composants, ou encore le prix d'un Bitcoin sur deux plateformes différentes). Si cette relation s'écarte de sa norme (par ex. un écart de prix anormal apparaît), le robot prendra des positions opposées (long sur l'actif sous-évalué, short sur le surévalué) en anticipant un retour à l'équilibre. **L'arbitrage classique** consiste à tirer profit d'un écart de prix d'un même actif sur des marchés différents : par exemple acheter un actif là où son prix est bas et vendre simultanément là où son prix est plus élevé. Par définition, **les opportunités d'arbitrage sont de courte durée** car le marché tend à corriger ces inefficiences rapidement. La stratégie doit donc être hautement automatisée et rapide pour en bénéficier [8] . Un principe de base est que les prix ne s'écartent pas très longtemps de leur valeur "juste" sans raison [9] . Dans le cas du robot, une approche d'arbitrage statistique peut être implémentée via un modèle de **cointégration** ou de **régression** entre deux actifs : par exemple, détecter si l'écart de rendement entre l'action A et l'action B dépasse 2 écarts-types de sa moyenne historique, ce qui déclenche un trade long A / short B. Sur cryptomonnaies, on pourrait surveiller les écarts de prix entre exchanges pour un même coin (arbitrage inter-exchange), voire des arbitrages triangulaires entre paires de devises crypto. L'arbitrage est généralement considéré comme une stratégie à **faible risque** (car neutre au marché si bien exécutée), toutefois les gains par transaction sont faibles ; le robot devra donc gérer les **coûts de transaction** avec attention et être capable d'exécuter un grand volume de trades rapidement pour accumuler les profits [10] . Enfin, cette stratégie étant basée sur la moyenne reversion (retour à la moyenne), elle complète bien la stratégie momentum (qui elle exploite l'écart qui se creuse) – en combinant les deux, le portefeuille peut gagner dans des phases de marché différentes.

- **Stratégie *Event-Driven* (basée sur les événements)** – Cette stratégie s'appuie sur l'**analyse des nouvelles et événements** pour prendre des positions avant ou peu après que le marché ait réagi. L'idée est d'exploiter les inefficiences de prix qui surviennent autour des annonces importantes [11]. Par exemple, une annonce de résultats trimestriels beaucoup meilleurs qu'attendu pour une entreprise peut entraîner une forte hausse de son action, mais parfois avec un léger délai ou de façon initialement désordonnée ; un système automatique peut détecter la nouvelle immédiatement, estimer son impact et prendre position avant que toute l'information ne soit pleinement *Pricée* dans le cours. Les types d'événements considérés incluent : publications de résultats et guidances, annonces de fusions-acquisitions, changements de direction, actualités macroéconomiques (taux d'intérêt, inflation), décisions réglementaires, ou même un tweet viral d'une personnalité influente (particulièrement pertinent en crypto). Pour ce faire, le robot intègre un composant **LLM d'analyse de news** (voir section suivante). Ce composant, couplé à une logique de décision, permettra de **classer chaque nouvelle en signal positif, négatif ou neutre** pour un ou plusieurs actifs. Par exemple, une news signalant une enquête judiciaire sur une entreprise serait classée "négative" pour l'action de cette entreprise et pourrait générer un signal de vente ou de couverture. Inversement, l'annonce surprise d'une baisse de taux par une banque centrale pourrait être classée "positive" pour les actions en général (et "négative" pour les obligations), déclenchant des ajustements de positions sur indices, etc. La stratégie event-driven se décline en sous-catégories plus spécialisées (merger arbitrage, trading sur annonces de résultats, etc.), mais dans le cadre de notre projet on se concentre sur la *réaction aux nouvelles en temps réel*. Cette stratégie est naturellement complémentaire des précédentes car elle apporte une **réactivité fondamentale** : même si aucune tendance technique ne se dessinait encore, une nouvelle majeure peut amorcer un mouvement que le momentum suivra ensuite. Le défi est d'éviter les faux signaux (par ex. news déjà anticipées ou rumeurs infondées), ce qui sera en partie géré par le LLM qui pourra filtrer la *signification* de l'actualité.

En combinant ces trois approches (momentum, arbitrage, event-driven), le robot pourra à la fois *suivre* les tendances établies, *corriger* les écarts injustifiés et *réagir* aux catalyseurs externes. Pour éviter les conflits entre stratégies, on définira des règles de **priorisation et de gestion de portefeuille** : par exemple, en situation normale chaque stratégie opère avec une allocation de capital dédiée et des limites de risque; en cas de signaux contradictoires (un arbitrage inciterait à acheter un actif tandis que le momentum le donnerait en vente), le système pourra soit s'abstenir, soit privilégier la stratégie ayant le plus haut score de confiance (par ex. une nouvelle très fortement négative pourrait primer sur un signal technique modérément haussier). Une approche plus avancée consistera à utiliser un **moteur d'orchestration** qui pèse les signaux de chaque stratégie pour formuler une décision finale optimisée (c'est-à-dire une sorte d'ensemble de stratégies, analogue à un comité de modèles).

Composant LLM d'Analyse des Nouvelles en Temps Réel

Un des éléments innovants du projet est l'intégration d'un **Large Language Model (LLM)** de type GPT, spécialisé dans le domaine financier, pour analyser les flux d'actualités en temps réel. Ce module, que l'on appellera le **LLM Analyzer**, jouera un rôle central dans la stratégie event-driven en **convertissant le langage naturel des news en signaux exploitables** par le moteur de trading. Voici comment il sera conçu et intégré :

- **Collecte et prétraitement des nouvelles** : Le pipeline de données enverra au LLM Analyzer chaque nouvelle pertinente dès sa réception (par ex. titre et résumé d'une dépêche Reuters, post sur un forum, etc.). Un prétraitement pourra inclure la suppression du bruit (nettoyage de HTML, correction des encodages), l'élimination des doublons, et éventuellement un **résumé automatique** si le texte est très long. Ce résumé peut être généré soit par une méthode

extractive simple, soit par un modèle de langue lui-même (p. ex. en utilisant une petite instance GPT pour condenser l'information). L'objectif est de présenter au LLM principal un texte concis qui capture l'essentiel de la nouvelle. (Dans certaines implémentations, on pourrait utiliser un modèle dédié pour résumer, tel que *Cohere's summarization model* choisi pour sa légèreté et son efficacité dans un projet similaire [12].)

- **Analyse par le LLM financier** : Le cœur du module est un modèle de langage de grande taille entraîné sur des données financières. On peut citer des exemples existants comme **FinBERT** (variante de BERT entraînée sur des textes financiers) ou **FinGPT** (LLM open-source orienté finance). Dans notre projet, nous pourrions utiliser une version fine-tunée d'un modèle type GPT-4 ou Llama, spécialisée dans le *sentiment analysis* financier. Ce modèle recevra le texte de la nouvelle et produira en sortie une **classification du sentiment/impact** – par exemple sous forme de score ou de probabilité {positif, neutre, négatif} pour chaque classe d'actifs d'intérêt. Concrètement, pour une nouvelle donnée, le LLM pourrait répondre que l'impact est « négatif 90% » pour l'action ABC, « neutre » pour le marché obligataire, « positif modéré » pour le secteur Énergie, etc. Pour simplifier, on se concentrera d'abord sur l'impact principal (positif/négatif) sur l'actif que le robot traite directement (par ex. l'action ciblée ou l'indice global). Il a été démontré que les LLM récents peuvent exceller à cet exercice de classification du sentiment appliqué aux textes financiers complexes [13]. Par exemple, le modèle **FinGPT** a été utilisé comme modèle principal de sentiment dans un système de trading, offrant une **classification fine (positive/négative)** des news avec des scores de confiance, grâce à un entraînement sur des données textuelles financières spécifiques [14]. L'utilisation d'un LLM permet de **capturer des nuances** dans le langage des communiqués économiques ou des discussions de marché (nuances souvent manquées par des analyseurs plus simples).

- **Intégration des scores de sentiment** : Les résultats du LLM Analyzer seront intégrés en temps réel au moteur de stratégie. Par exemple, si une nouvelle est classée « fortement négative » pour un actif, la stratégie event-driven émettra immédiatement un signal de vente ou de réduction de position sur cet actif. Ces signaux basés sur les nouvelles pourront aussi influer sur les autres stratégies : un signal négatif fort pourrait temporairement suspendre une position momentum haussière (éviter d'acheter juste avant une mauvaise nouvelle), ou au contraire un signal positif très marqué pourrait accélérer l'ouverture d'une position avant que la tendance ne reflète la nouvelle. Techniquement, le **score de sentiment** du LLM sera traité comme une variable additionnelle dans le système de décision. On peut imaginer un **module de fusion** qui prend en entrée les indicateurs techniques et prédictions quantitatives d'une part, et les alertes du LLM d'autre part, afin de produire une décision finale.

- **Performance et maintenance** : L'intégration d'un LLM en temps réel pose des défis de performance (temps d'inférence) et de mise à jour. Pour assurer la réactivité, on pourra déployer le modèle sur une machine performante (GPU) ou utiliser une version optimisée/réduite du modèle. Par exemple, FinGPT a été conçu avec une approche *LoRA* (Low-Rank Adaptation) qui permet d'affiner un LLM avec un surcoût computationnel réduit [15], ce qui est intéressant pour un déploiement efficient. On veillera également à mettre à jour régulièrement le LLM (ou à le *fine-tuner* en continu avec de nouvelles données, si possible) pour qu'il reste au fait du langage et des situations financières récentes. Dans un premier temps, un modèle pré-entraîné suffira (éventuellement validé sur un ensemble de news historiques dont on connaît l'impact réel sur les marchés, afin de vérifier la qualité de ses classifications).

En résumé, le LLM Analyzer agit comme les « yeux et oreilles » du robot sur l'actualité financière. Là où les modèles quantitatifs et techniques voient les chiffres, le LLM voit la *raison* derrière ces chiffres (nouvelles, psychologie du marché). **Combiner ces deux niveaux d'analyse** offre un potentiel

d'information supérieur pour générer des signaux de trading. Des projets de recherche récents ont d'ailleurs démontré la valeur ajoutée d'une approche combinée : un système qui synthétise les données de marché en temps réel et le sentiment extrait des nouvelles par un LLM peut produire des signaux de trading plus précis et actionnables [13].

Architecture Technique du Système

L'architecture proposée pour ce robot de trading est **modulaire** et conçue pour être réalisable avec des technologies courantes. L'accent est mis sur la séparation des responsabilités (ingestion des données, analyse, exécution) et sur l'évolutivité (possibilité d'ajouter de nouvelles stratégies ou de nouvelles sources de données). Voici une vue d'ensemble de l'architecture, du flux de données et des composants :

[16] *Architecture modulaire combinant données de marché, analyse par LLM et signaux de trading.*

- **Flux de données entrants** : D'un côté, le **flux marché** (prix/volumes) entre dans le système via le module *Market Data Handler*. De l'autre, le **flux de nouvelles** entre via le module *News Handler*. Ces deux modules sont responsables de se connecter aux sources externes (exchanges, APIs de news), de normaliser les données et de les publier en interne (par ex. via des messages ou événements). Chaque nouveau tick de prix ou nouvelle importante est encapsulé dans un message enrichi (avec timestamp, symbole de l'actif, etc.) et diffusé aux composants consommateurs.

- **Stockage et Historique** : Un **Data Store** central (ou plusieurs spécialisés) accumule les données : il stocke à haute fréquence les prix reçus (pour servir de fenêtre glissante en intraday et pour archive historique) et conserve également un journal horodaté des nouvelles et des analyses produites (par ex. on peut stocker chaque nouvelle avec le score de sentiment calculé). Ce stockage peut être implémenté avec une base de données timeseries (comme InfluxDB ou Timescale) pour les ticks de marché, et une base NoSQL ou relationnelle pour les données de news et signaux. L'historique alimente l'entraînement des modèles hors-ligne, tandis qu'en live il sert de **mémoire** courte pour les calculs de moyennes mobiles, etc.

- **Module d'Analyse Technique & Pricing** : Ce composant traite principalement les données de marché. Il calcule en temps réel les **indicateurs techniques** requis par les stratégies momentum (ex. moyennes mobiles, RSI, oscillateurs, etc.), actualise les modèles quantitatifs de pricing (par ex. recalcul d'une juste valeur via un modèle factoriel si certaines variables de marché ont bougé) et exécute les modèles prédictifs ML entraînés (en passant en entrée les dernières données disponibles pour obtenir, par exemple, une prévision de rendement à l'horizon 1h ou 1 jour). Ce module peut être subdivisé en *sous-modules* par stratégie ou par type de modèle. À chaque pas de temps, il fournit un **état du marché analysé** : tendances actuelles, éventuels signaux bruts (par ex. « tendance haussière forte détectée » ou « écart de 3% par rapport au prix théorique »).

- **Module LLM Analyzer (Analyse des nouvelles)** : Décrit en détail dans la section précédente, il fonctionne en parallèle de l'analyse technique. Chaque nouvelle entrant déclenche (après prétraitement) une inférence du modèle de langage, produisant un **score de sentiment** ou un étiquetage de l'événement. Ce module peut fonctionner de façon asynchrone (les news arrivent à des intervalles imprévisibles), mais il transmet immédiatement ses résultats aux autres composants dès qu'ils sont prêts. On peut envisager une petite file d'attente pour les nouvelles si elles arrivent en rafale, afin de les traiter séquentiellement ou avec plusieurs instances du LLM en parallèle si nécessaire.

- **Moteur de Stratégie (Strategy Engine)** : C'est le **cerveau décisionnel** du robot. Il agrège les informations provenant de l'Analyse Technique/Pricing et du LLM Analyzer. Sur la base de règles prédéfinies et de paramètres configurables, il décide d'**ouvrir, fermer ou ajuster des positions**. On peut implémenter le moteur de stratégie comme un ensemble de sous-modules, un par stratégie, orchestrés par un module central :

- Le sous-module Momentum examine les indicateurs techniques et éventuellement les prédictions ML de tendance pour générer des **signaux de trade directionnels** (achat/vente).
- Le sous-module Arbitrage surveille les paires d'actifs ou les flux multi-marchés pour détecter des **écarts de prix** et génère des ordres long/short couplés.
- Le sous-module Event-Driven attend les **alertes du LLM** et déclenche des ordres rapides basés sur l'actualité (par ex. vendre immédiatement X unités de l'actif A sur une news négative).

- Un module d'**agrégation des signaux** collecte ensuite toutes les propositions de trades des sous-modules. Il applique des règles de priorité ou de fusion (par ex. si deux stratégies donnent des ordres contraires sur le même actif, résoudre le conflit en fonction de la hiérarchie ou de la confiance). Il en résulte une liste finale d'ordres à passer.

- **Module d'Exécution et Gestion d'Ordres** : Ce composant prend les décisions du moteur de stratégie et les envoie sur le marché. Il interagit avec les API de trading du courtier ou de l'exchange, en respectant les protocoles (envoi d'ordres limit/market, gestion des confirmations, etc.). Ce module doit être conçu pour être **réactif et fiable** : il intègre des contrôles de risque *en dernière ligne* (par ex. vérification qu'une position totale n'excédera pas un seuil, ou qu'un ordre de vente ne dépasse pas la quantité détenue, etc.) avant exécution. Idéalement, l'Order Manager comprend un **système de Risk Management** qui effectue des contrôles globaux (limites de perte quotidienne, exposition maximale par actif, etc.) [17] [18] . Ainsi, même si le moteur de stratégie propose un ensemble d'ordres agressifs, le module d'exécution peut en bloquer une partie pour rester dans le cadre de risque défini. L'exécution étant automatisée, on pourra optimiser ce module pour la latence (pour l'arbitrage notamment, la rapidité est cruciale). Après envoi, les confirmations d'ordres et mises à jour de positions sont renvoyées au moteur de stratégie pour mise à jour de l'état interne (boucle de feedback).

- **Interface de Monitoring / Dashboard** : Pour présenter ce projet en entretien, on peut aussi proposer une interface de visualisation (même sommaire). Elle afficherait en temps réel l'état du robot : les **prix en streaming**, les **indicateurs clés** calculés, les **scores de sentiment** des dernières nouvelles, et les **décisions de trading** prises (par ex. un log des ordres passés, P&L en temps réel, etc.). Cela permet de démontrer le fonctionnement du système de façon transparente. Techniquement, cette interface se connecte aux différents modules (ou à un bus central) pour extraire les données d'état.

Du point de vue déploiement, l'architecture peut être réalisée via un ensemble de micro-services communicants (par exemple, un service pour l'ingestion des données, un pour le LLM, un pour chaque stratégie, etc., orchestrés éventuellement par Kubernetes pour la scalabilité [19]), ou de façon monolithique organisée en threads/tâches asynchrones si on préfère la simplicité. L'important est de garantir la **scalabilité** (pouvoir ajouter facilement une nouvelle stratégie ou une nouvelle source de données), la **résilience** (si le module LLM tombe en panne, le reste du système continue à fonctionner en mode dégradé, etc.) et la **faible latence** entre la réception d'un événement de marché et l'émission d'un ordre.

Modules Clés du Système

Pour clarifier l'organisation du projet, voici un résumé des principaux modules composant le robot de trading, avec leur rôle respectif :

- **Module d'Ingestion de Données** – Gère la **connexion aux flux externes** (données de marché en temps réel via API/broker, flux de nouvelles via API/RSS). Il normalise et stocke les données brutes et les distribue aux autres modules. *Exemple*: connexion WebSocket à un exchange pour recevoir en continu les derniers prix, ou appel périodique d'une API de news.

- **Module d'Analyse Technique & Pricing** – Regroupe les fonctions de **calcul d'indicateurs** (momentum, volatilité, moyennes, etc.), d'**évaluation quantitative** (valorisation théorique, modèles financiers) et éventuellement les **modèles ML prédictifs** entraînés. Il transforme les données brutes de prix en insights exploitables (tendance haussière, signal de sur-achat, prix théorique vs actuel, prévision de rendement, etc.).

- **LLM Analyzer (Analyse des Nouvelles)** – Composant LLM basé sur un modèle de langage de type GPT, spécialisé finance. Il **analyse chaque nouvelle textuelle** et en produit une **classification de sentiment/impact** (positif, neutre, négatif) accompagnée d'un score de confiance. Ses sorties alimentent la stratégie événementielle. (On peut implémenter ce module via une API vers un service GPT ou héberger un modèle open-source comme FinGPT localement).

- **Strategy Engine (Moteur de Stratégies)** – Coeur décisionnel où sont implémentées les **stratégies de trading**. Il contient plusieurs sous-modules (Momentum, Arbitrage, Event-Driven, etc.) qui génèrent des signaux ou ordres candidats en fonction des données analysées. Un orchestrateur interne agrège ces signaux et prend les **décisions finales** de trading en respectant les règles de gestion (priorités, évitement de doublons ou conflits, etc.).

- **Module d'Entraînement & Backtesting** – Environnement hors-ligne pour **entraîner les modèles** (par ex. ajuster le modèle ML sur l'historique, ou affiner le LLM si besoin) et **tester les stratégies** sur des données passées. Ce module permet de valider la performance du robot avant sa mise en production et d'effectuer des ré-entraînements réguliers. Il utilise la base de données historique et peut simuler les décisions du Strategy Engine sur ces données afin de calculer les métriques de performance (Sharpe ratio, drawdown, taux de réussite des trades, etc.).

- **Module d'Exécution & Gestion d'Ordres** – Responsable de l'**interface avec le marché réel**. Il convertit les décisions du Strategy Engine en ordres effectifs (achat/vente) via l'API du courtier ou de l'exchange, et assure le suivi des ordres jusqu'à leur exécution. Inclut des fonctionnalités de **Risk Management** intégrées (par ex. annuler ou réduire un ordre si le risque cumulé dépasse une certaine limite, gestion des stop-loss automatiques).

- **Monitoring & Logging** – Bien que souvent transversal, il mérite d'être mentionné : l'ensemble du système doit **journaliser** les actions (données reçues, décisions prises, ordres exécutés, erreurs éventuelles) et offrir des outils de **visualisation** en temps réel. Cela se matérialise par un tableau de bord ou une console, utile pour la présentation (montrer comment le robot réagit en live) et pour le développement (debugging, analyse post-mortem des trades).

Chaque module peut être développé et testé de manière relativement indépendante, ce qui facilite le travail en équipe et l'itération. En entretien, on pourra présenter un **schéma d'architecture** illustrant ces modules et leurs interactions (par exemple, une vue type pipeline de gauche à droite : Sources de données -> Analyses -> Stratégies -> Exécution). On insistera sur la **cohérence de l'ensemble** : les données historiques alimentent les modèles d'analyse; les analyses (techniques et textuelles) alimentent les stratégies; les stratégies pilotent les ordres; et le tout forme une boucle auto-apprenante (les nouveaux résultats de marché peuvent re-calibrer les modèles sur la durée).

Conclusion

En synthèse, ce projet de robot de trading algorithmique s'articule autour de choix stratégiques clairs : **(1)** une classe d'actifs appropriée (actions pour la richesse des données et l'impact des news, ou cryptomonnaies pour l'innovation et la volatilité exploitable), **(2)** un pipeline de données robuste mêlant historique et temps réel pour alimenter le système en continu, **(3)** des techniques sophistiquées de pricing et de prévision (modèles quantitatifs et machine learning) fournissant des signaux de base, **(4)** plusieurs stratégies de trading avancées (momentum, arbitrage, event-driven) travaillant de concert pour capturer différentes opportunités de marché, et **(5)** une intégration pionnière d'un LLM financier analysant les nouvelles en temps réel pour donner une dimension supplémentaire à la prise de décision. L'architecture modulaire proposée permet de **maîtriser la complexité** en isolant chaque fonction clé du système, tout en assurant leur interaction fluide via des flux de données bien définis. Un tel projet, même présenté de façon simplifiée, démontre une compréhension approfondie des domaines de la finance de marché, du génie logiciel et de l'intelligence artificielle. C'est un **sujet concret et ambitieux** à présenter en entretien, témoignant à la fois de compétences techniques (développement d'un système distribué temps réel, implémentation d'algorithmes de trading, utilisation de l'IA) et de la capacité à résoudre un problème métier complexe (améliorer la performance de trading grâce à la technologie). En conclusion, ce plan donne une feuille de route pour discuter de la réalisation d'un robot de trading de nouvelle génération, associant le meilleur des deux mondes : la *quant finance* traditionnelle et les dernières avancées en *IA générative*.

Sources Utilisées : Les idées et justifications s'appuient sur des références reconnues, par exemple la nécessité d'une approche multi-stratégie pour améliorer le profil risque/rendement [1], la profitabilité du momentum sous conditions [7] notamment sur des marchés volatils [2], la définition classique de l'arbitrage [20] et son automatisation indispensable [8], l'exploitation d'événements d'actualité pour générer des trades [11], ou encore l'intégration réussie de l'analyse du sentiment via LLM dans des systèmes de trading récents [13] [14]. Ces éléments étayent la crédibilité du projet et pourront être cités en appui lors de l'entretien.

---

[1] Multi-Strategy Portfolios: Combining Quantitative Strategies
https://blog.quantinsti.com/multi-strategy-portfolios-combining-quantitative-strategies-effectively/

[2] 5 Crypto Trading Strategies to Ride Momentum
https://www.osl.com/hk-en/academy/article/5-crypto-trading-strategies-to-ride-momentum

[3] [12] [13] [14] [15] [16] [19] An End-To-End LLM Enhanced Trading System
https://arxiv.org/html/2502.01574v1

[4] [5] [6] Machine Learning for Stock Prediction: Solutions and Tips
https://www.itransition.com/machine-learning/stock-prediction

[7] [8] [9] [10] [20] Automated Trading Strategies: How It Works | IT Craft
https://itechcraft.com/blog/all-about-automated-trading-strategies/

11 Event-Driven Trading Strategies (Event-Based Trading - Backtest Insights) - QuantifiedStrategies.com
https://www.quantifiedstrategies.com/event-driven-trading-strategies/

17 18 Automated Trading Systems: Architecture, Protocols, Types of Latency
https://blog.quantinsti.com/automated-trading-system/