CrypTrade

Investir dans les cryptos n'a jamais été aussi facile ! Le Bitcoin, l'Ethereum, le Monero, tous à portée de main !

Lyon 2021
EPITA

BUTHOD Victor
victor.buthod@epita.fr

MASSON Jacques
jacques.masson@epita.fr

MAYER Julien
julien.mayer@epita.fr

FAURE Joris
joris.faure@epita.fr

Table des matières

Première partie

Introduction

CrypTrade est un logiciel de trading de crypto-monnaie qui dans un premier temps, récupére la valeur réelle de différentes crypto-monnaies à partir d'une API (sur ce site par exemple), afin de créer un simulateur conforme au cours réel des cryptos-monnaies. Vous pouvez acheter et vendre ces monnaies manuellement, et prochainement vous pourrez le faire automatiquement via une intelligence artificielle intégrée.

1 Qu'est-ce qu'une crypto-monnaie ?

À l'inverse des monnaies "physiques" comme l'Euro, la Livre ou encore le Dollar, émises et régulées par les gouvernements, une crypto-monnaie est une monnaie virtuelle, dont l'émission et l'évolution ne dépendent pas d'un état ou d'une banque centrale mais qui repose sur l'ensemble de ses utilisateurs.
On peut citer notamment le Bitcoin, mais aussi l'Etherum, le Monero, le Lite Coin, le Doge Coin, etc.

Toutes les crypto-monnaies reposent sur ce que l'on appelle la Blockchain (littéralement la "chaine de blocs").
Pour comprendre ce dont il s'agit, il faut vous représenter la Blockchain comme un immense livre de comptes dans lequel sont inscrites toutes les informations des transactions de crypto-monnaies, celles-ci ne sont alors pas vérifiées par une banque mais par tout un réseau d'utilisateurs, les mineurs.
La Blockchain contient une trace de l'intégralité des échanges réalisés jusqu'ici. Elle est non seulement indestructible, mais surtout consultable à tout moment par n'importe qui.

2 Principe du trading

La stratégie principale du trading est la suivante :
L'objectif est d'acheter une monnaie avant que la valeur de celle-ci augmente et de la vendre juste avant qu'elle ne baisse.

Avec l'avènement du trading algorithmique sur les places financières mondiales, de nouvelles technologies voient le jour dans les institutions financières. En effet, les programmes informatiques peuvent être bien plus efficaces que n'importe quel trader pour analyser les données et réagir au bon moment.

3 Notre algorithme

C'est là que commence la partie algorithmique du projet. Un bot sera intégré au logiciel, nous convertirons différentes stratégies de trading en algorithme afin que notre bot achète ou vende des monnaies à des moments optimisés.

En plus de l'algorithme de base, nous implémenterons plusieurs fonctionnalités additionnelles qui permettent d'optimiser les transactions du bot de la même manière que le ferait un vrai trader de Wall Street.

Deuxième partie
L'équipe

4 Victor Buthod

Pour cette deuxième période de soutenance, j'ai d'abord créé un historique nous permettant d'accéder aux anciennes valeurs des cryptomonnaies, afin de pouvoir faire des calculs dessus. J'ai décrit en détail le procédé utilisé dans la partie **Historique**. Puis j'ai ensuite restructuré intégralement le projet, car il était mal organisé, et donc peu pratique, j'ai décris ma méthode en détail dans **Restructuration**. Et enfin, je me suis occupé de toute la partie création et implémentation du graphique, avec laquelle j'ai fait face à beaucoup de problèmes expliqués dans la partie **Graphique**.

5 Joris Faure

Durant cette deuxième phase de développement, j'ai continué ma tache de responsable de l'interface graphique et de suppléant sur les fonctionnalités additionnelles.
Après notre première soutenance nous avions les boutons Buy, Sell d'implémenté. Pour cette deuxième soutenance il a donc fallu implémenter les fonctionnalités de Stop-Loss et de Levier à notre interface.
Dans un premier temps j'ai donc repris le code de Julien et de son Stop-Loss afin de l'implémenter. Pour ce faire, j'ai utilisé Glade pour créer une case où l'utilisateur pourra entrer la valeur du Stop-Loss qu'il voudra appliquer. Ensuite j'ai implémenté un bouton qui permettra d'appliquer ce Stop-Loss. Lorsque l'utilisateur modifie la case d'entrée du Stop-Loss, la fonction "on_sl_entry_changed" s'exécutera et récupérera en temps réel la valeur entrée par l'utilisateur. Lorsque l'utilisateur appuiera sur le bouton "Place stop-loss", la valeur récupérée sera ensuite traité par la fonction de Julien "on_sl_button_clicked" que j'ai modifié afin qu'elle puisse aussi afficher dans notre interface graphique le Stop-Loss

appliqué actuellement. En effet la valeur récupérée sera réécrite dans le label de l'interface "Pending Stop-Loss".
Après avoir implémenté la fonctionnalité Stop-Loss à l'interface, je me suis concentré sur le système de Levier qui multiplie les gains et les pertes de l'utilisateur. J'ai donc commencé par essayer d'appliquer une méthode naïve qui consiste à multiplier la valeur qui contrôle les gains d'argent par la valeur du levier (x1/x10/x50/x100). Seulement vu que la variable qui contrôle les gains était aussi relié au bouton buy et Sell, le levier multipliait les achats et les ventes. J'ai alors mis en place un algorithme qui multipliait uniquement les gains effectués par l'utilisateur. Malheureusement cet algorithme augmentait de façon exponentielle les gains de l'utilisateur. Julien et Jacques ont donc repris les bases de mon implémentation du levier afin de complètement le retravailler et le faire marcher.
Enfin, j'ai débuté la création du Musk Detector. Pour cela je dois accéder à l'API de twitter afin de récupérer les tweets d'Elon Musk. Seulement, pour accéder à l'API de Twitter il faut en faire la demande à l'entreprise en expliquant notre projet afin qu'ils valident ma requête. Je suis toujours dans l'attente d'une réponse en espérant l'avoir d'ici peu afin de commencer à réellement travailler dessus !

6 Julien Mayer

Pour cette deuxième soutenance, je me suis occupé des fonctionnalités additionnelles tels que le Stoploss et l'effect de Levier.

Pour le Stoploss, j'ai ajouté une variable global "sl" qui enregistre la dernière valeur entrée par l'utilisateur dans le cadre de l'interface. Cette variable est une chaine de caractère qu'on convertira par la suite en float.
J'ai ensuite ajouté un paramètre limit dans la structure Money qui correspond à la valeur du Stoploss pour une monnaie. Ainsi, chaque monnaie peut avoir un Stoploss différent.
La limite de la monnaie actuellement affichée est mise à jour à l'appel de la fonction "on_sl_button_clicked". Pour savoir sur quelle structure il faut la modifier, j'utilise la variable "on_money" qui indexe la monnaie actuellement affichée sur l'interface.
L'application du Stoploss se fait dans la fonction "on_money_possess" servant à actualiser l'évolution du nombre d'une crypto-monnaie qu'on possède en dollars "usd_possess". Après l'avoir mise à jour grâce au nouveau prix d'une crypto-monnaie, je vérifie que le Stoploss est activé sur cette monnaie et si "usd_possess" devient inférieur à la limite fixée, alors il faut vendre toutes les unités de la monnaie concernée. Pour cela, il m'a suffi d'enregister dans "amount" la valeur "usd_possess" et d'appeler la fonction "on_sell_button_clicked" afin de simuler une vente par l'utilisateur. Il ne reste plus qu'à remettre le Stoploss de la crypt-monnaie à 0 pour le désactiver.

Pour le système de levier, j'ai retravaillé sur les fonctions que Joris avait faite avant car nous avons remarqué que les calculs n'étaient pas justes. Avec l'aide de Jacques nous avons repris entièrement son fonctionnement de la manière suivante :
Nous avons créé une variable globale par monnaie afin d'enregister un levier par crypto-monnaie. Son initialisation est similaire à celle du Stoploss à la différence que nous n'avons pas encore intégré le levier dans la structure Money. Nous avons ensuite modifié le calcul du nouveau "usd_possess" dans "on_money_possess".
Maintenant, à chaque actualisation, la nouvelle valeur est égale à :

argent investi + (unité possédé x nouveau prix - argent investi) x levier

J'ai aussi rangé la plupart des variables globales que nous avions créées en plusieurs exemplaires pour les differentes monnaies (comme btc_usd_possess) dans la structure pour plus de clarté.

7 Jacques Masson

Pour cette deuxième soutenance, j'ai été responsable du site et suppléant sur les fonctionnalités additionnelles. Pour le site, ce dernier a été mis à jour. Il ne nous manque plus que la page des réseaux, mais vu que le site n'est pas le plus important j'ai préféré aider Joris et Julien sur les fonctionnalités additionnelles. Pour ces dernières, après la refonte de la structure money et le début de la fonction levier, j'ai pu finaliser cette dernière pour qu'elle marche comme on le veut. Enfin je me suis renseigné sur les stratégies du Bot avec mes équipiers pour que l'on trouve la meilleure stratégie pour notre intelligence artificielle.

Pour le bouton du levier, avec Julien nous avons créé trois variables pour chacune des monnaies, cela permet à l'utilisateur de choisir un levier différent pour chacune de ses monnaies. De plus comme l'a dit Julien nous avons changé le calcul du levier car le précédent était faux et nous donné les résultats d'une suite géométrique. Enfin nous avons décidé que les leviers de base qui était les leviers x1, x2, x5, x10 étaient trop petit et nous apportait pas assez de possibilités. Nous avons donc choisi finalement des leviers x1, x10, x50, x100, ce qui nous permet de voir la différence quand on active les leviers sans que cette ne soit trop grande.

Troisième partie
Notre projet

8 Fonctionnalité Additionnel

8.1 StopLoss

Pour cette deuxième soutenance, nous avons implémenté le Stoploss permettant d'éviter des pertes trop importantes lorsque la valeur d'une monnaie achetée passe sous une limite fixée par l'utilisateur. Nous avons ajouté à la structure Money un paramètre limit afin que chacune de nos crypto-monnaies disponible sur le simulateur puis avoir un Stoploss qui lui est propre. Lorsqu'une valeur est entrée dans le champ correspondant sur l'interface, et elle est stockée dans une variable globale sous la forme d'une chaine de caractères. Après ça, elle est convertie en flottant et affectée à la bonne structure grâce à la variable globale "on_money" correspondant à la monnaie actuellement affichée sur l'interface.

Lorsqu'on fait varier la quantité en dollars d'une monnaie possédé "usd_possess", on vérifie que le Stoploss a été activé sur cette monnaie. Si c'est le cas, et que "usd_possess" passe sous la limite fixée, alors on vend la totalité de ce qu'on a sur cette monnaie en appelant la fonction on_sell_button_graphe_toggle() et la limit est remise à 0.

8.2 Levier

Pour cette deuxième soutenance nous avons ajouté à notre logiciel, une des fonctionnalités les plus connues dans les logiciels de crypto-monnaies, le principe d'effet de levier.
L'effet de levier est une technique de margin-trading, un moyen simple d'augmenter son capital d'investissement. Plus risqué, il permet de réaliser des gains plus importants, mais aussi d'amplifier ses risques de pertes.
Le levier consiste à emprunter des capitaux sur la plateforme d'échange. Cette dernière, vous prête des capitaux, par rapport à votre investissement initial. Il est généralement possible d'investir avec un levier sur un long ou un short.

En d'autres termes il s'agit d'un multiplicateur qui va jouer sur les gains et les pertes, on peut alors gagné plus en investissant moins mais aussi perdre beaucoup plus. Par exemple :
-Je souhaite investir 100€ sur le Bitcoin. Si j'applique un levier x5 alors, si la valeur du bitcoin augmente de 5%, je gagnerais 5x5% : 25€ sur 100€ de base au lieu de 5€. A l'inverse, si le Bitcoin perd 5% de sa valeur, je perds 25€ sur mes 100€.
-J'investis 100€ sur le Bitcoin avec un levier x5. Le Bitcoin perd 20% de sa valeur, 20x5 = 100%. Ma position est donc liquidé, j'ai perdu mes 100€.

Dans notre logiciel l'effet de levier se présente de la manière suivante : l'utilisateur pourra choisir entre quatre leviers différents, un levier x1, un levier x10, un x50, et un x100. De plus ces leviers sont propres à chaque monnaie, on peut par exemple avoir un levier x10 pour le bitcoin, un levier x50 pour l'ethereum, et un levier x100 pour le dogecoin.
Pour ce faire nous nous avons initialisé 3 types de variable une pour chaque levier, ces variables se présentent de la sorte, "int btc_lev" , "int eth_lev" et "int doge_lev". Ensuite avec ses 3 variables et la variable "on_money", l'utilisateur pour choisir le levier qu'il veut utiliser pour la monnaie sur laquelle il travaille.
Ensuite la fonction qui va mettre à jour la variable "btc->usd_possess" s'appelle "void on_money_possess(int i_money)", cette fonction va faire de la manière suivante :

```
1   case 0:
2    printf("USD possess = %f, nb_possess = %f\n",
3        btc->usd_possess, btc->nb_possess);
4    init_pos = btc_init_pos;
5    newPrice = btc->nb_possess*btc->priceUsd;
6    lvlEffectPrice = (newPrice-init_pos)*btc_lev;
7    btc->usd_possess = init_pos+lvlEffectPrice;
8
9    if (btc->limit > 0 && btc->usd_possess < btc->limit)
10   {
11       btc->limit = 0;
12
13       sprintf(amount, "%f", btc->usd_possess);
14       change_crypt_amount(btc);
15   }
16   sprintf(array, "%f : %f$",
17           btc->nb_possess, btc->usd_possess);
18   gtk_label_set_text(GTK_LABEL(btc_possess),
19                                       (gchar*)array);
```

Exemple de la fonction "on_money_possess" pour le bitcoin

Comme on peut le voir à la ligne 4 la variable "init_pos" va récupéré le nombre de bitcoin en dollars au moment de notre dernier achat. la variable "newPrice" elle va calculer la valeur du nombre de bitcoin que l'on possède en fonction de son prix à l'instant T. Ensuite ligne 6, la variable lvlEffectPrice va recevoir la valeur de la différence entre la variable "newPrice" et "init_pos" mulitplié par le levier que l'on a choisi. Enfin, on incrémente la valeur de "btc-> usd_possess" par la variable "lvlEffectPrice".

```
struct Money
{
  char          *id;
  int           rank;
  char          *symbol;
  char          *name;
  float         supply;
  float         maxSupply;
  float         marketCapUsd;
  float         volumeUsd24Hr;
  float         priceUsd;
  float         changePercent24Hr;
  float         vwap24Hr;
  float         usd_possess;
  float         nb_possess;
  float         limit;
  struct Money  *next;
};
```

Example d'une structure avec le bitcoin

9 Interface du Simulateur

9.1 Graphique

Pour créer le graphique représentant la variation des valeurs des monnaies, nous avons premièrement essayé de faire un programme qui créait les graphiques à partir de deux listes de coordonnées x et y, puis le transformait en une image en . *png*. Nous y sommes parvenu en utilisant ce projet trouvé sur github.

Mais nous avons pensé que créer une image à chaque nouvelle actualisation des valeurs était couteux, donc nous avons décidé de partir sur une autre stratégie ; afficher directement le graphique dans Glade grace à la bibliothèque SDL.

Nous avons donc créé un tout nouveau dossier afin de recommencer la partie graphique depuis le début avec SDL. Nous avons divisé le nouveau fichier *SDL_ graph.c* en cinq fonctions principales.
La première remplissant entièrement la page avec la couleur blanche.
La deuxième créant les lignes d'abscisse et d'ordonnée en arrière-plan afin de pouvoir se repérer sur le graphique.
La troisième, la plus importante, faisait en sorte de récupérer les valeurs de la liste de l'historique du prix des monnaies *hist*, et traçait ensuite sur le graph des droites reliant deux coordonnées "x" et "y" ("x" étant le temps, et "y" les valeurs des monnaies), à "x+5" (l'ancien temps auquel on ajoute 5 secondes) et "y+1" (la prochaine valeur des monnaies).
La quatrième fonction servait simplement à attendre que l'utilisateur ferme la fenêtre SDL, pour garder le graphique affiché aussi longtemps que désiré.
Et enfin, la dernière fonction se chargeait d'initialiser les fenêtres à la bonne taille, et d'appeler les autres dans le bon ordre.
Le nouveau dossier graph fonctionnait parfaitement, bien que nous étions un peu dessus car nous pensions gagner des performances mais au final, nous n'avons pas notifié de changement dans le temps d'exécution.

Il nous restait ensuite à intégrer ce nouveau graphique à glade, pour cela nous avons trouvé deux extensions "GtkGlExt" et "Gtksdl", malheureusement, GtkGlExt ayant été fermé en 2007, elle est peu documentée, et n'est compatible qu'avec des versions de Glade antérieur à la nôtre.
De plus, Gtksdl est encore plus vieille, elle date de 2001 environ. Nous avons tout de même essayé d'incorporer SDL à Glade, mais après des heures de recherche sans résultat concluant, nous avons décidé de revenir à l'ancienne méthode infiniment plus simple.

L'ancien fichier créant un graphique et l'enregistrant sous la forme d'une image ayant déjà été créée, il nous a simplement suffi de le lier à l'historique qui avait été crée entre-temps, puis de l'incorporer à glade à l'aide d'une simple fonction permettant de changer une image à partir de son nom qu'on appelle à chaque nouvelle actualisation des valeurs des monnaies.

9.2 Boutons

9.2.1 Le Stop-Loss

Pour l'implémentation graphique de la fonctionnalité Stop-Loss, nous avons créé une case d'entrée utilisateur en bas de l'écran. L'utilisateur entrera une valeur et pourra ensuite la valider en appuyant sur le bouton "Place STOP-LOSS". La valeur entrée par l'utilisateur est récupéré par notre programme en temps réel dans la variable "amount" puis lorsqu'il appuiera sur le bouton, il exécutera la fonction associée et pourra visualiser le Stoploss validé en bas à droite de l'écran. Selon la monnaie sélectionnée sur l'interface, la fonction citée précédemment va mettre à jour la valeur du paramètre limit dans la structure de la monnaie précédemment sélectionnée. Celle-ci est toujours connue grâce à une variable globale on_money tel que : BTC = 0, ETH = 1, DOGE = 2 La variable limit correspond à la valeur minimum autorisée de "usd_possess". Si on passe en dessous de cette valeur, le Stoploss va vendre toutes les unités de la monnaie concernée.

9.2.2 Le bouton levier

Le levier peut être sélectionné par l'utilisateur en cochant l'un des quatre boutons visibles sur l'interface. Les valeurs du levier disponible sont 1, 10, 50 et 100, ce qui est similaire aux autres outils de trading disponibles sur internet.
De même que pour le Stoploss, en cliquant sur l'un des boutons on va appeler l'une des fonctions correspondantes. Cette fonction va alors affecter la valeur séléctionnée à la structure Money de la monnaie actuellement affichée par l'interface. Comme à chaque fois, cette dernière est indexée par la variable globale on_money.

10 Startègie du bot

10.1 Rencontre avec un spécialiste

Nous avons discuté via téléphone avec un étudiant en écoles de commerce pratiquant le trading depuis plusieurs années. Après lui avoir détaillé notre projet et les outils à notre disposition, il nous a expliqué le fonctionnement du trading et différentes stratégies utilisées par les vrais traders. Certaines techniques dont ils nous à parler seront trop dur à implementer pour notre bot car la fréquence des données que nous récupérons est assez faible et nous n'avons pas tout les outils nécessaires. Mais nous avons pu retenir plusieurs options applicables à notre projet. Il nous reste maintenant à choisir la stratégie la plus adaptée avant de pouvoir créée le Bot.

10.2 Les stratégie de trading

Le Day Trading est un style selon lequel les traders achètent et vendent plusieurs monnaies dans une seule séance de trading, clôturant souvent leurs positions jusqu'à la fin de la journée. En fait, c'est un phénomène extrêmement rare pour les traders actifs de tenir des positions du jour au lendemain.
Les échelles de temps les plus couramment utilisées dans les stratégies de day trading sont de quatre heures, une heure, trente minutes et quinze minutes.

Le Swing Trading est une méthode par laquelle les traders achètent et vendent des monnaies dans l'optique de les conserver pendant plusieurs jours et, dans certains cas, même des semaines.
Pour cette méthode, utilisent souvent le graphique journalier pour passer des ordres qui sont généralement conformes à la tendance générale du marché.
L'une des techniques les plus populaires pour le swing trading est l'utilisation des indicateurs de trading. Il existe plusieurs types d'indicateurs sur le marché et ils présentent tous à la fois des avantages et des inconvénients.

Le trading saisonnier implique la possibilité d'avoir une tendance annuelle répétable. De nombreux marchés ont souvent des caractéristiques saisonnières en raison des modèles répétitifs, tels que les conditions météorologiques, les annonces macroéconomiques du gouvernement et les bénéfices des entreprises.
Un trader saisonnier pourrait utiliser ces modèles saisonniers comme une sorte d'indicateur pour orienter ses transactions. En conséquence, même si le trading saisonnier n'est pas un système d'achat ou de vente, il peut cependant offrir au trader un contexte plus large.

10.3 Historique

Pour réaliser l'historique, nous avons ajouté un pointeur *next* qui pointe, à chaque nouvelle récupération de la valeur d'une monnaie, vers la précédente. C'est le même principe que pour une liste chainée. Grâce à cela, nous avons créé une fonction *get_history* qui parcourt toutes les structures en récupérant leur prix en dollars, et crée dynamiquement une chaine d'entiers *hist* disponible depuis tous les fichiers du projet. Un entier *hist_len* qui indique la longueur de la liste *hist* est créé en même temps pour ne pas avoir à parcourir toute la liste à chaque fois que nous voulons récupérer la longueur de l'historique.

11 Problèmes rencontrés

11.1 Problème levier

Dans le calcul du nouveau "usd_possess", nous utilisions l'ancien "usd_possess" à la place de l'argent investi. Avec le calcul défini de cette manière, la différence étant multipliée par le levier, nous nous retrouvions avec une suite géométrique de raison du levier. Alors la quantité d'une monnaie possédée en dollars augmentés considérablement aux files des actualisations. Ce calcul n'était pas réaliste car la variation sur "usd_possess" doit être proportionnel à la variation du prix d'une monnaie. Nous avons donc du enregister l'argent investi au moment de l'achat pour pouvoir modifier le calcule.

11.2 Restructuration

Pour la première soutenance, nous avions placé la fonction *main* qui exécute notre programme dans le fichier *UI.c*. Or, lors de cette deuxième période de soutenance, nous avons eu besoin d'appeler des fonctions de *UI.c* depuis d'autres fichiers. Mais créer un header au fichier comprenant la fonction *main* n'est pas pratique. Nous avons donc dû restructurer l'entièreté du projet afin d'être plus efficace.
Pour cela, nous avons déplacé notre *Makefile* dans le dossier racine de 667-investments, et créé un fichier *main.c* à cet endroit.
Pour récupérer les variables des structures d'une monnaie, nous utilisions la fonction *get_struct* qui lisait les fichiers texte contenant les informations sur les monnaies à chaque appel, et cela dans chaque fonction où nous en avions besoin. Ce n'était absolument pas optimisé, nous avons donc changé complètement de méthode.
Désormais, chaque cryptomonnaie à une structure à son nom, qui est initialisée comme variable globale directement dans le fichier *fetcher.c* (là où nous actualisons les fichiers texte contenant les informations sur les monnaies). La fonction *get_struct* est appelée directement après l'actualisation. De cette façon, nous pouvons accéder aux structures des monnaies depuis tous les fichiers sans même avoir à appeler de fonctions tiers.

Quatrième partie
Site internet et Réseaux

12 Site internet

Pour la réalisation du site internet nous allons coder en HTML car cela nous permet de créer un site assez simple que nous compléterons avec du code CSS pour embellir le résultat. Le site sera composé de plusieurs pages sur lesquels l'utilisateur pourra se déplacer.

La première page sera l'accueil, cette page regroupe des photos de notre projet ainsi qu'une partie avec les actualités nouvelles et anciennes. Comme deuxième page, nous avons la page Groupe, cette page est dédiée à une courte présentation du groupe et des personnes qui le compose pour réaliser le projet. La page réseaux sociaux, elle servira à donner tous les liens vers nos pages Twitter, Instagram, et autres si nous en trouvons l'utilité.

La page FAQ, elle, est faite pour permettre à nous, les développeurs, d'interagir et de répondre aux questions des utilisateurs, et ainsi leur permettre une meilleure expérience de l'application. La page suivante est la page des liens utiles, dans cette page nous avons inscrit nos différents emails, et aussi les différents des ressources que nous avons utilisées.

Il y aura la page chronologie de la réalisation où nous raconterons toutes les étapes de la réalisation du projet de ses débuts jusqu'à sa finalité. Enfin il y aura une page de téléchargement du logiciel où l'utilisateur pourra télécharger le logiciel sous plusieurs versions différentes afin qu'il puisse choisir la version qu'il désire. D'autres pages arriveront si nous pensons qu'elle est nécessaire pour l'utilisateur et ses demandes, notamment une page sur le projet en lui-même, c'est-à-dire nos motivations à le faire, ce qui le rend unique par rapport aux autres, et toutes les étapes de sa conception.

Le site a été mise à jour et nous avons travaillé sur la qualité de ce dernier pour le rendre plus attrayant, et ainsi le faire ressembler à un vrai site d'entreprise comme on peut le voir sur les sites les plus connus.

Le site actuel de CrypTrade

13 Réseau sociaux

Pour la promotion de notre application, nous avons décidé de lui créer des pages dédiées sur les différents réseaux sociaux tels que Instagram ou encore Twitter. Cela permettra aux utilisateurs de suivre l'avancement de notre application. Aussi ces pages nous serviront à interagir avec les différents utilisateurs, recevoir leurs critiques afin d'améliorer l'application, ou encore de créer des tutoriels pour les débutants dans l'application.
Pour l'instant les utilisateurs peuvent découvrir les différentes améliorations et ajouts sur notre compte Twitter.

Cinquième partie
Répartition des charges

	Répartition des tâches				
	Interface graphique	**Stratégie du bot**	**Récupération des données**	**Fonctionnalités additionnelles**	**Le site**
Joris	Resp.	Supp.		Supp.	
Victor	Supp.	Supp.	Resp.	Resp.	
Julien		Resp.	Supp.	Supp.	
Jacques	Supp.	Supp.			Resp.

Sixième partie
Conclusion

En conclusion pour ce deuxième rapport de soutenance, nous avons bien avancé ce projet que nous voulons tous mener à bien, le fait que l'équipe soit aussi soudée permet les interactions entre les membres plus efficaces. Pour résumer l'avancée du projet, nous avons implémenté l'historique des valeurs, ce qui nous permet de créer une base de données que le bot analysera à l'avenir.
Nous avons ajouté deux fonctionnalités additionnelles, le Stoploss et le levier, ce qui complète notre simulateur de trading.
Pour ce qui est de l'interface, nous avons ajouté les champs nécessaires au fonctionnement des fonctionnalités citées précédemment ainsi que l'affichage des Stoploss actifs.
De plus, le centre de notre interface contient maintenant un graphique de l'évolution des valeurs qui s'actualise en temps réel.
La plupart des erreurs de segmentation de la dernière soutenance ont été résolu. Pour ce qui est de l'avancement général du projet, nous avons respecté ce que nous avions prévu de faire pour cette deuxième soutenance, nous ne sommes donc pas en retard quant à l'avenir du projet. Il nous reste encore à faire tous les algorithmes pour les différentes stratégies que CrypTrade sera capable d'adopter.

Septième partie
Annexes

lien vers Youtube : `https://www.youtube.com/`
lien vers StackOverflow :`https://stackoverflow.com/`
lien vers Gold Price : `https://goldprice.org/`
lien vers le compte Twitter : `https://twitter.com/667investments`