

|  |
| --- |
| EL HADDADI Yassine EL BAKKOURI Yassine KASSABEH Zakariya JABBOUR Hanâ |

|  |
| --- |
| Projet informatique |
| 2018-2019 |



Table des matières

[1. Introduction 2](#_Toc11540952)

[2. Fonctionnement 3](#_Toc11540953)

[2.1 Page d’accueil 3](#_Toc11540954)

[2.2 Interface client (Customer) 4](#_Toc11540955)

[2.3 Interface vendeur (Seller) 7](#_Toc11540956)

# Introduction

Dans le cadre de l’unité « Projet informatique » nous avons eu l’opportunité de mettre en pratiques les différences connaissances et compétences acquises lors du premier quadrimestre en matière de génie logiciel, bases de données et enfin, programmation orientée objet.

Le but de ce projet était la prise en main d’un projet informatique de l’analyse des besoins du client à la conception et enfin au développement d’un application permettant la vente d’armoires en kit. Ainsi est né KitBox.

Dans un premier temps, nous avons d’abord dû analyser les besoins du client et donc nous imprégner de l’énoncé. Nos professeurs faisant office de client, nous avons dû à plusieurs reprises retourner vers eux et les questionner quant à leurs attentes afin d’être sûrs de les satisfaire au mieux.

Ensuite, nous fîmes place à la réflexion en ce qui concerne l’implémentation du code et la stratégie à adopter. Pour cela, les diagrammes UML ont été un outil précieux pour documenter nos choix d’architecture et tenter de les exposer le plus clairement aux clients.

Une fois cela de réaliser, l’implémentation du code se profila petit à petit, au gré des différentes réunions faites avec le client afin que, sur base d’une interface, celui-ci nous fasse part de leur approbation ou remarques éventuelles

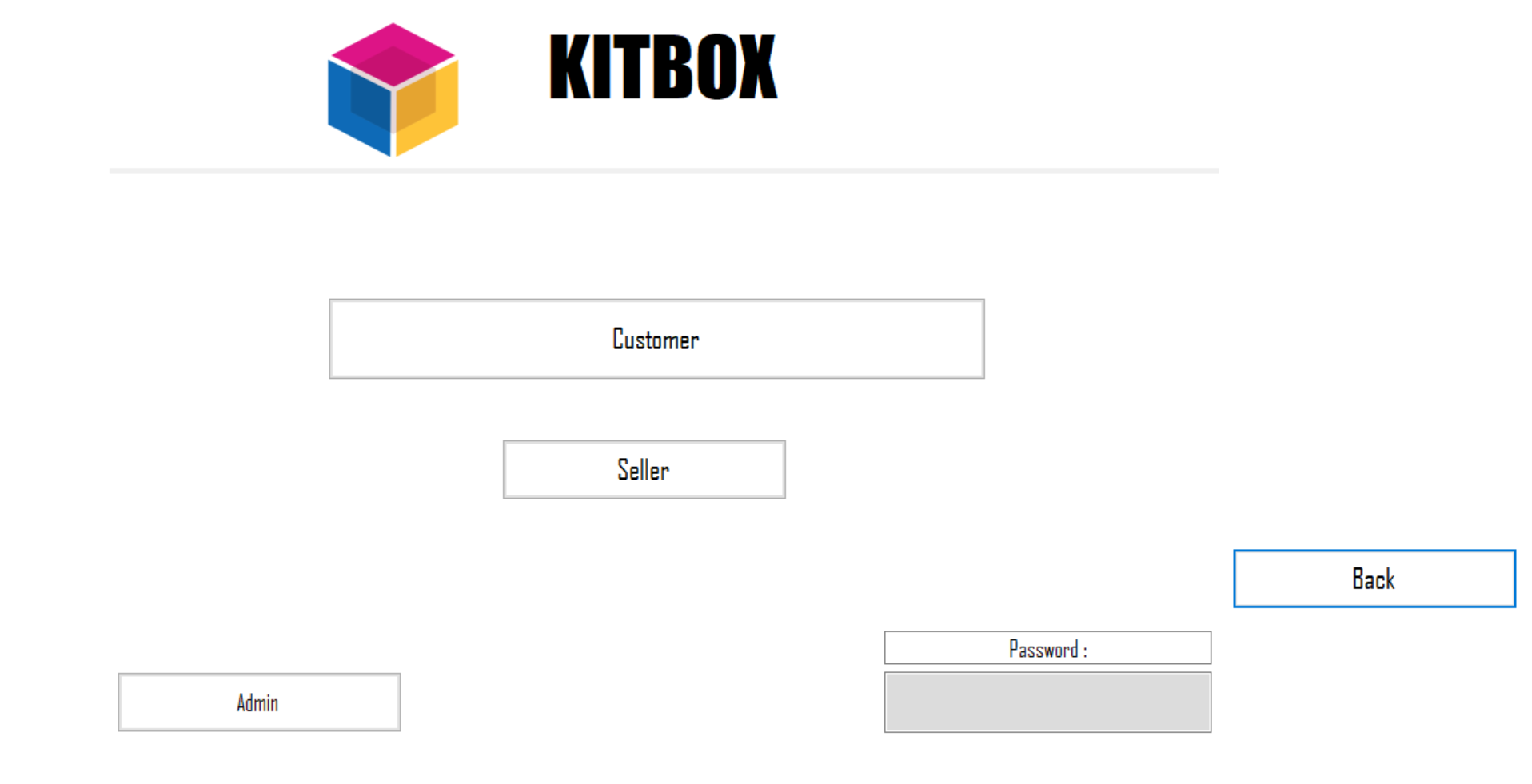
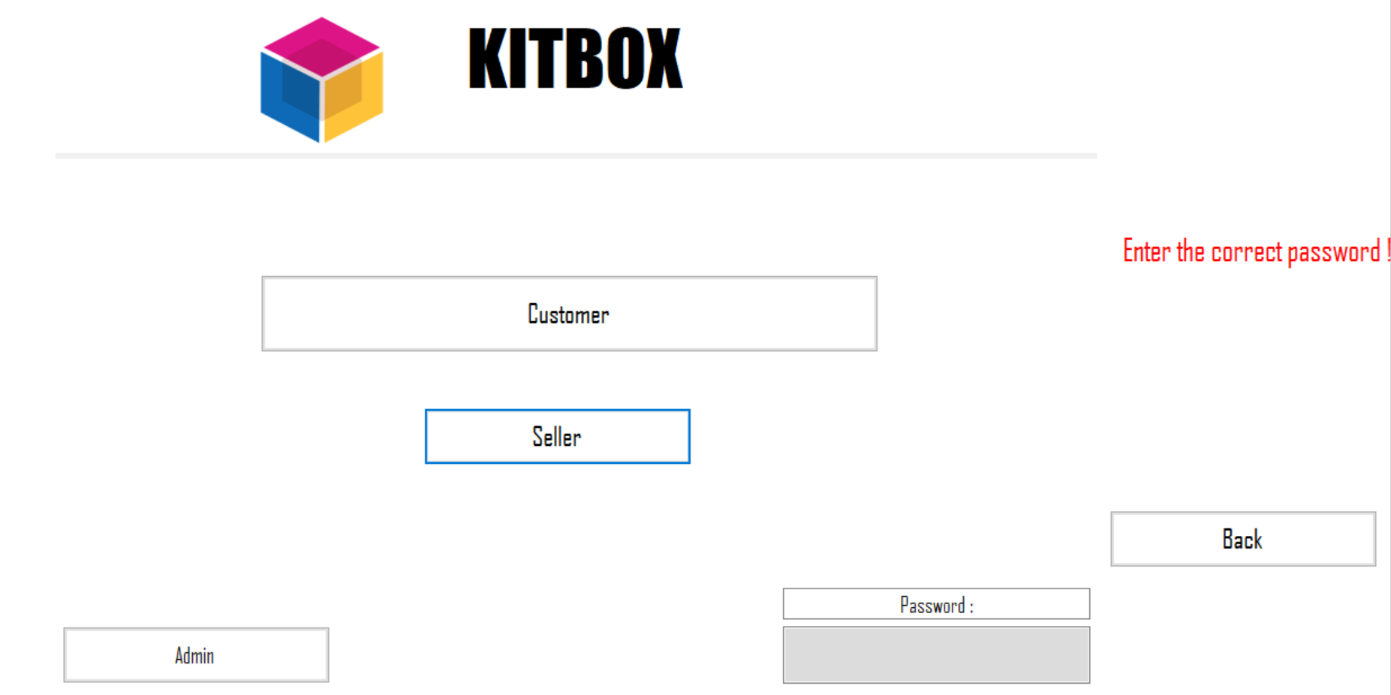
Ce rapport présentera lieu le fonctionnement de l’application en premier lieu suivi d’une description des différents choix technologiques et en guise de conclusion, une discussion du bilan des objectifs et de la collaboration dans le groupe.

# Fonctionnement

L’application KitBox a été développée en langage de programmation C# via le logiciel Visual Studio. Concernant la base de données nécessaire pour la sauvegarde des commandes, stocks et client, elle a été réalisée MySQL sous phpMyAdmin. Elle est gratuite et en ligne afin que les différents membres du groupe puissent aisément y avoir accès et travailler dessus sans devoir toujours la rectifier manuellement sur chaque pc.

## Page d’accueil

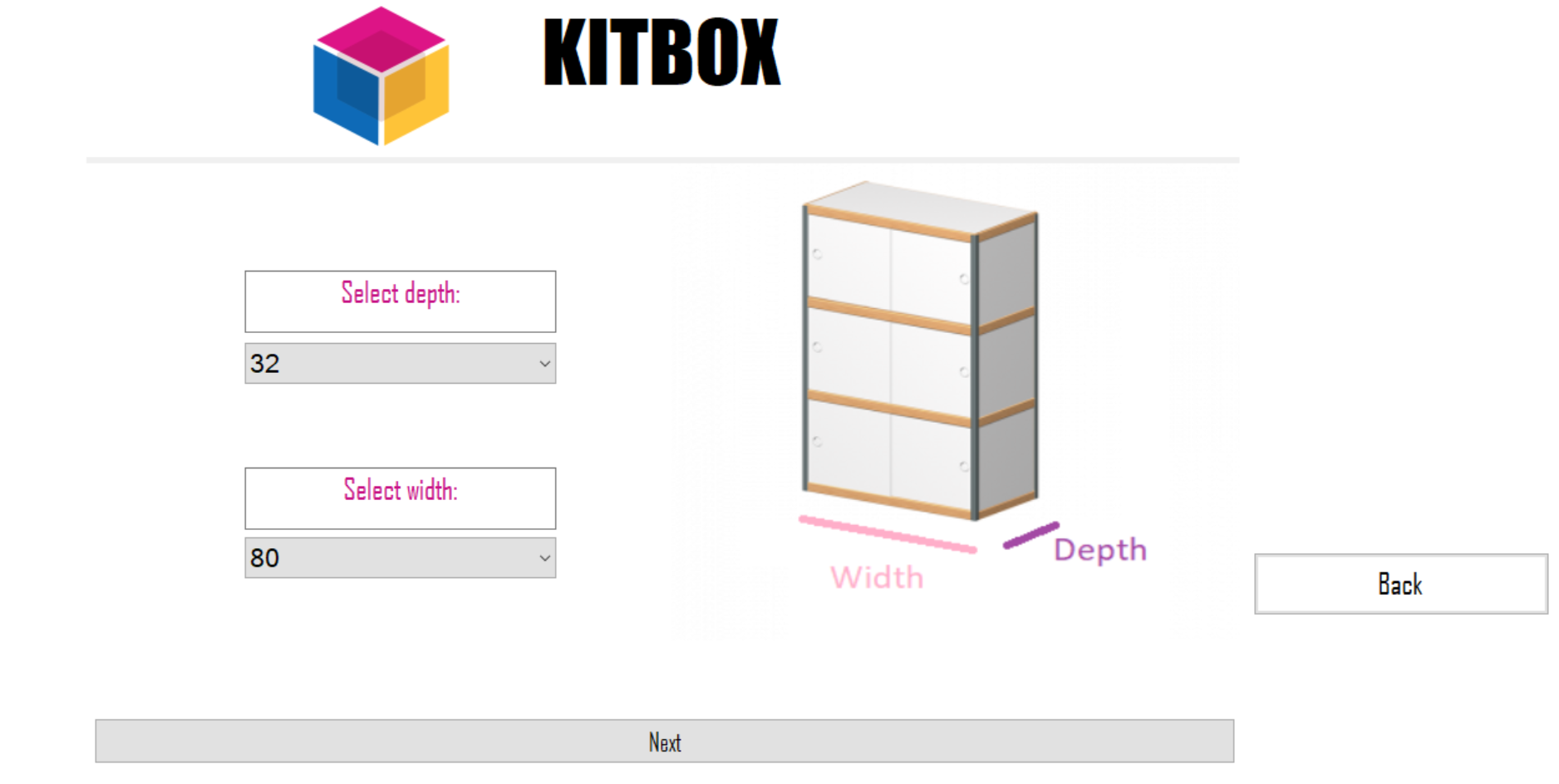
Lors de son ouverture, l’application se présente comme ceci :

Il y a trois boutons principaux : le bouton « **Customer** » permettant d’accéder à l’interface client où une commande ou un devis pourront être effectués, le bouton «**Seller** » permettant d’accéder à l’interface réservée au vendeur et enfin le bouton « **Admin** », donnant accès à l’interface magasinier où celui-ci peut accéder aux différentes commandes. Les deux dernières interfaces sont accessibles seulement lorsque le bon de mot de passe, « 0000 », est entré. Dans le cas échéant, un message stipulant d’enter le bon de mot de passe apparaitra :

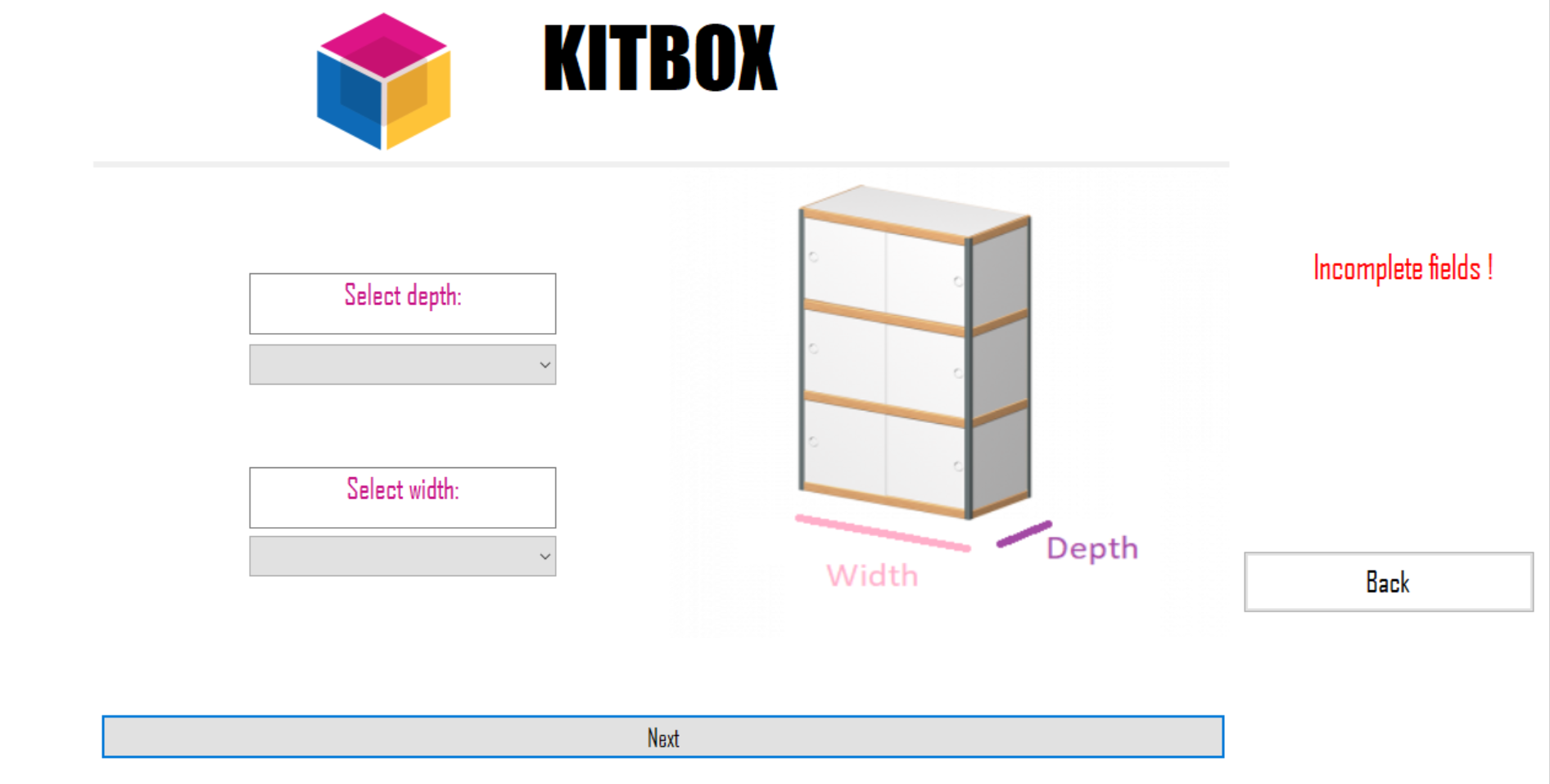
Enfin, le bouton « **Back**» permet de revenir en arrière et ainsi naviguer entre les différentes interfaces.

## Interface client (Customer)

Lorsqu’un client arrive au magasin, il ne peut avoir accès qu’au bouton « Customer » et ainsi à l’interface client. Celle-ci se présente comme suit :

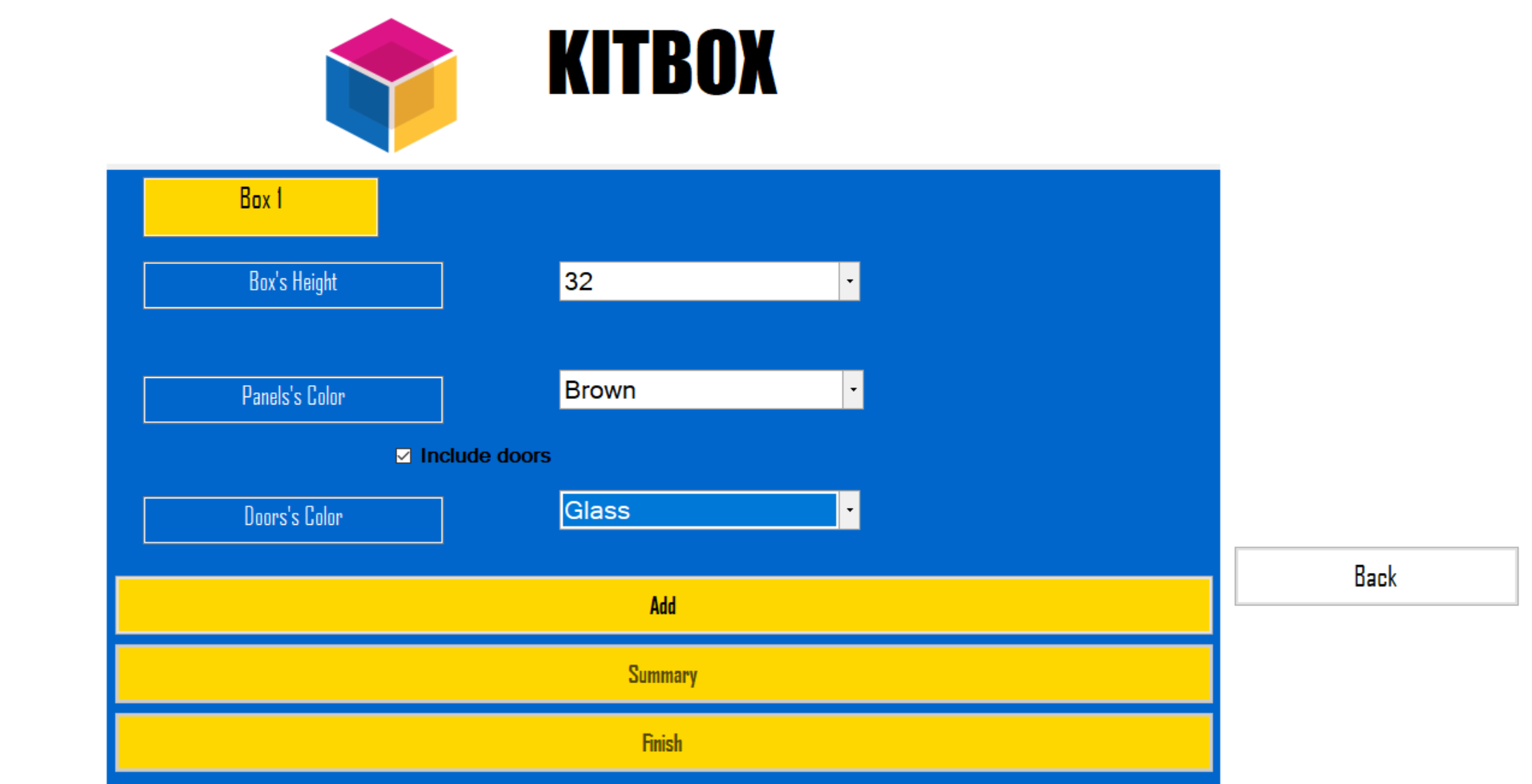


La première chose est de définir la profondeur et la largeur de l’armoire car celles-ci sont communes pour tous les casiers.  
Si ces champs sont incomplets, l’utilisateur ne pourra pas aller plus loin et un message d’erreur apparaitra :

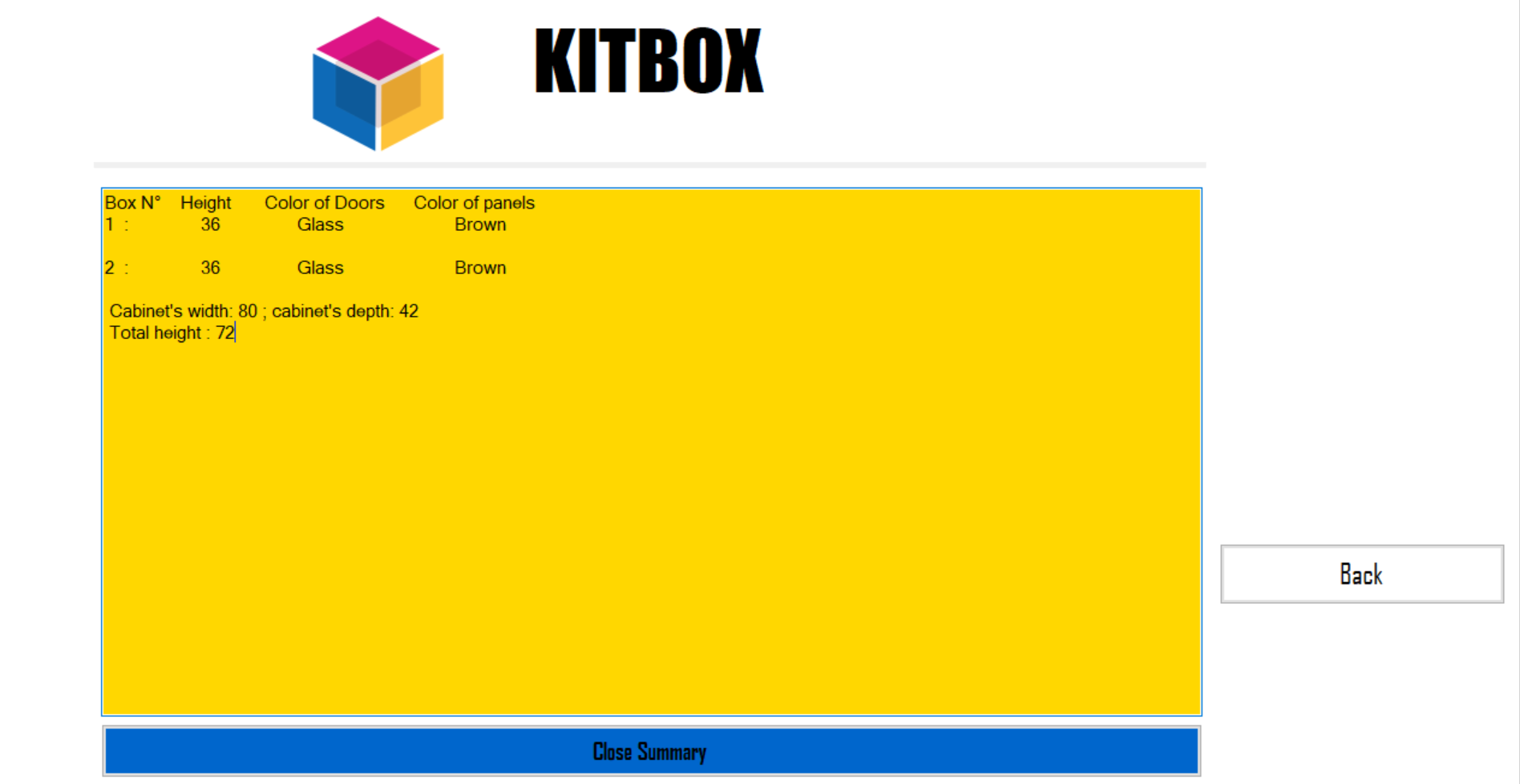


L’utilisateur peut revenir à tout moment sur la page d’accueil à l’aide du bouton « **Back**».

Une fois la largeur et la longueur sélectionnées, l’utilisateur accède à une interface lui permettant de choisir, au gré de ses envies, les casiers au gré qui composeront son armoire. Ainsi, il pourra sélectionner une hauteur, une couleur et préciser s’il désire ou non une porte.  
A noter que pour certaines largeurs, il n’est pas possible d’inclure des portes au casier. La notion « Doors available » est indiquée à côté de valeurs permettant l’ajout de portes dans le menu déroulant.

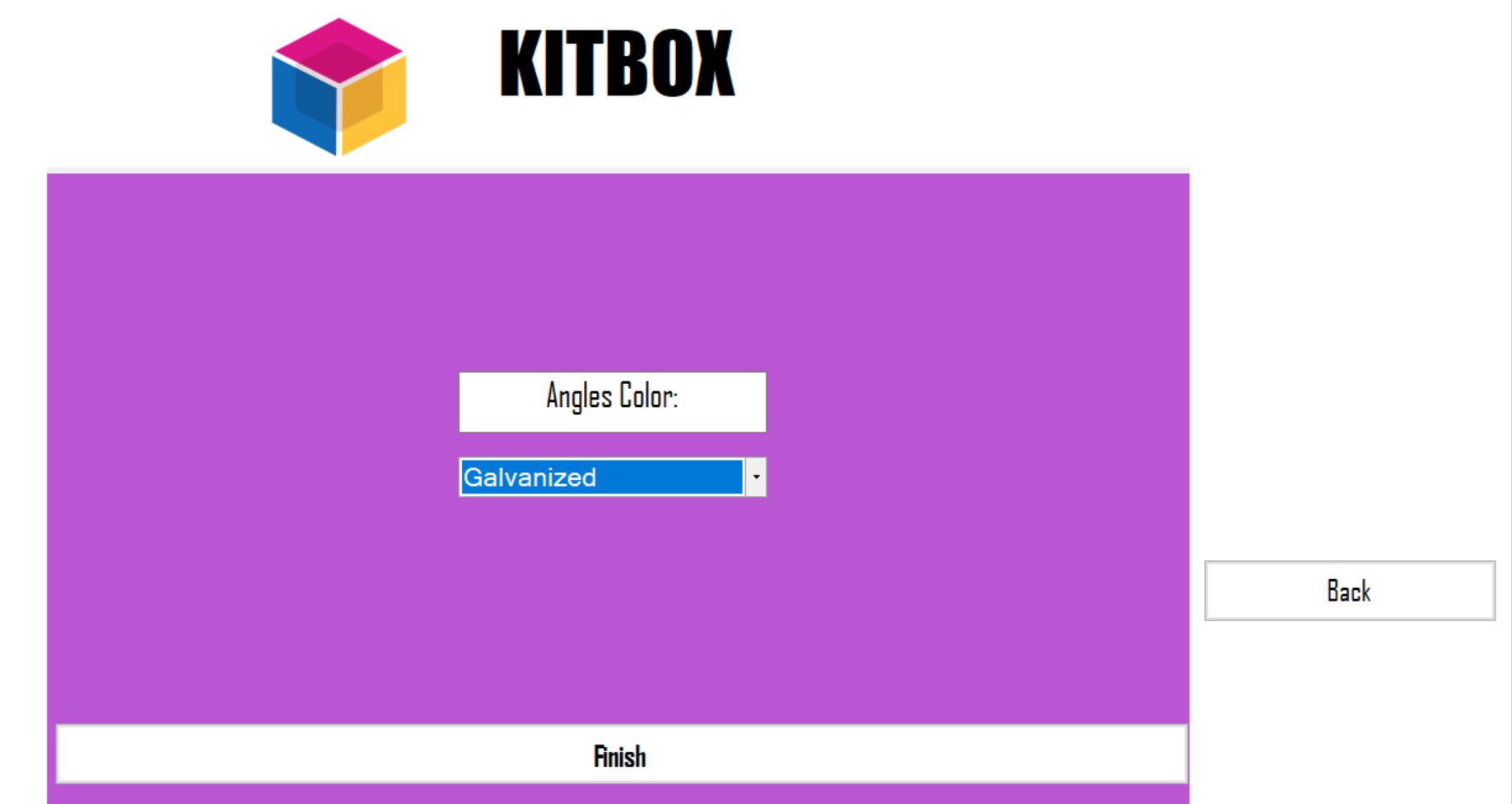


Le bouton **« Add »** permet d’ajouter le casier défini et passer au suivant. Le nombre de casiers est limité à 7 selon le désir du client.  
Le bouton **« Summary »** permet d’avoir un résumé de la commande à tout moment lors de la création de l’armoire.

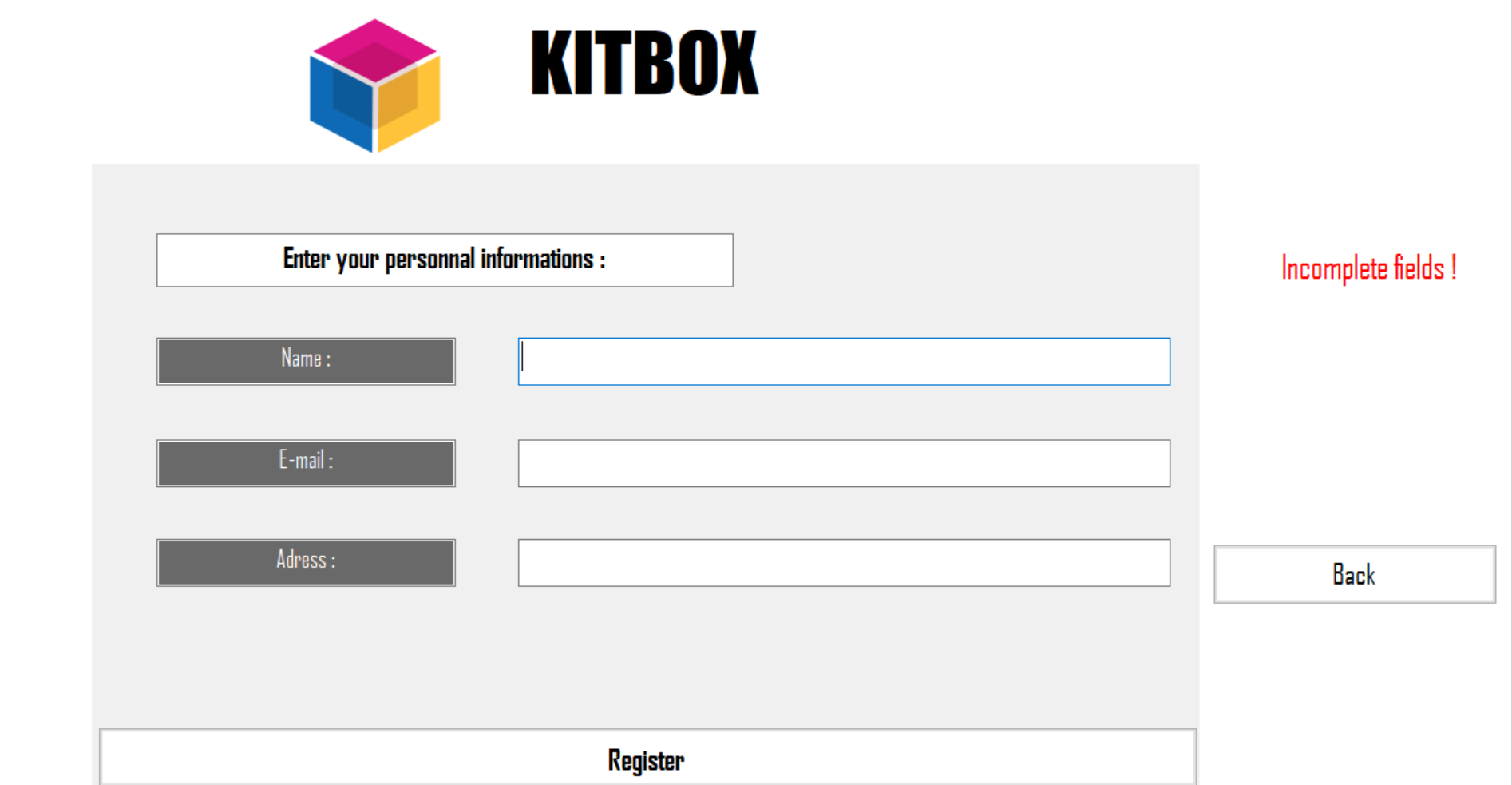


Enfin, le bouton **« Finish »** permet de finir la commande.

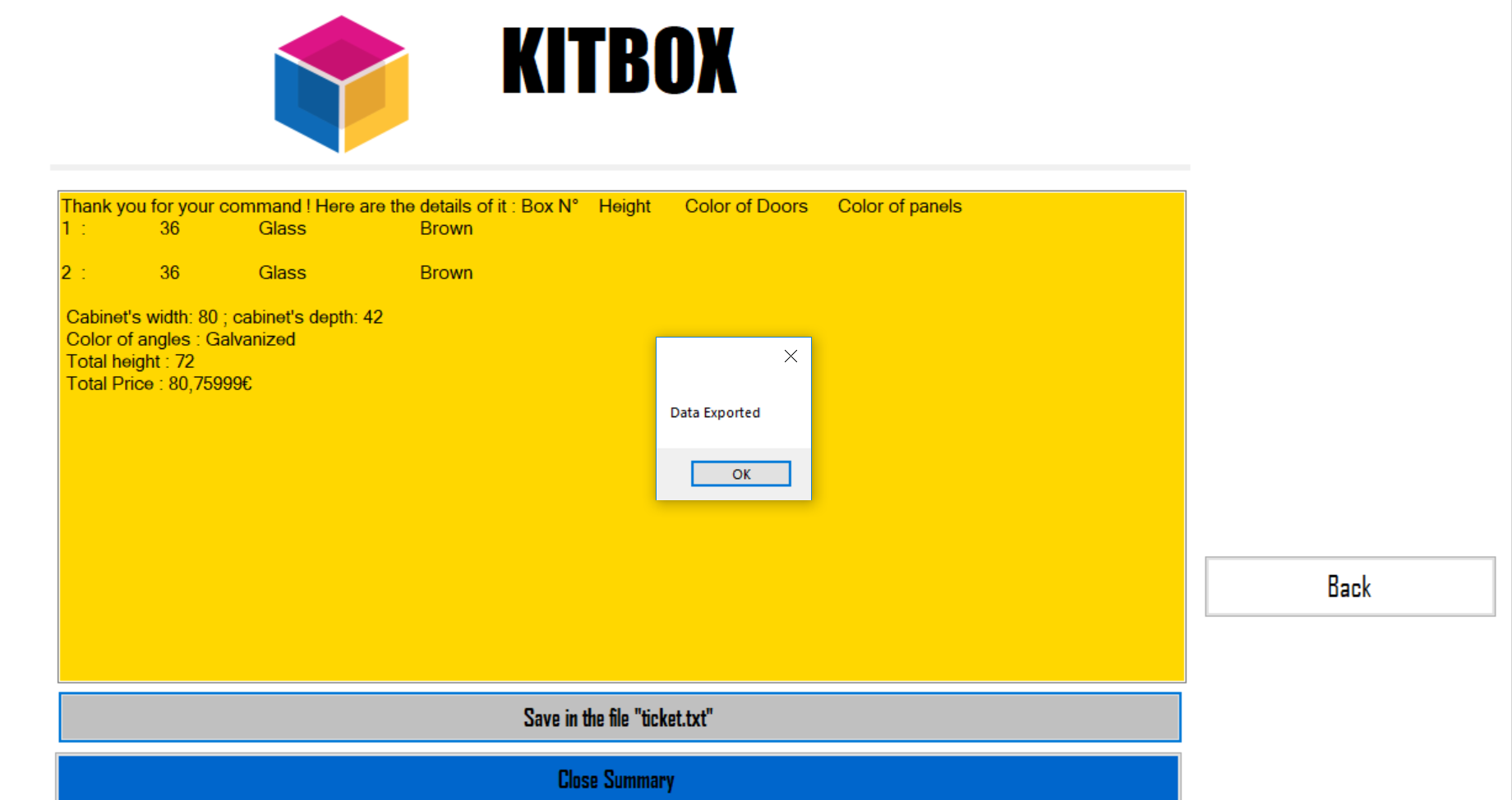
Une interface permettant le choix de la couleur des cornières apparait lorsque la commande est clôturée.



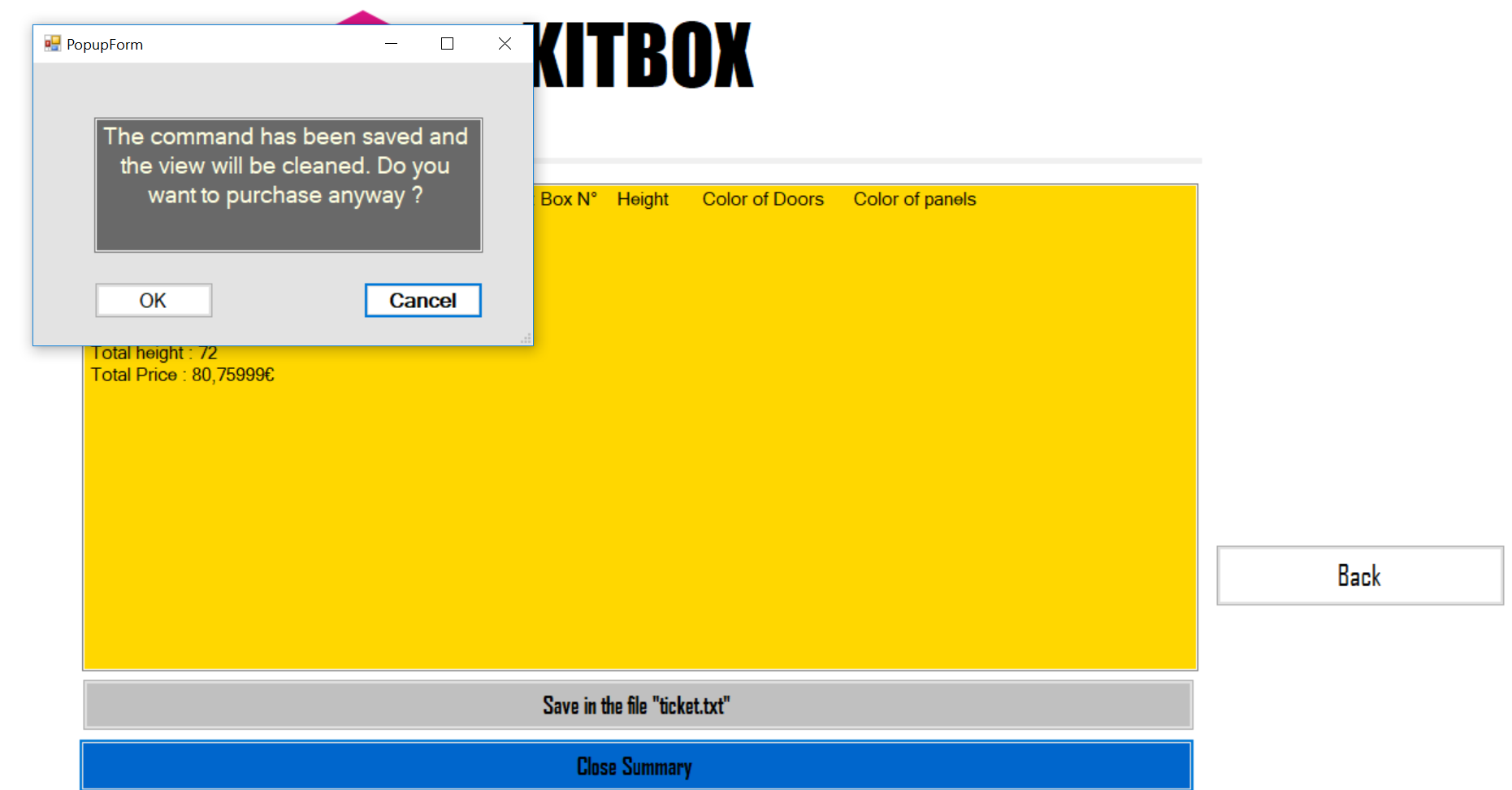
Lorsque le choix est fait et confirmé à l’aide du bouton « Finish », l’utilisateur doit entrer ses informations personnelles et s’enregistrer afin de finaliser la commande. Dans le cas contraire, un message d’erreurs apparaitra.



Une fois enregistré, un récapitulatif avec le prix total apparait et une fenêtre pop-up signale au client que ses données ont bien été enregistrées.



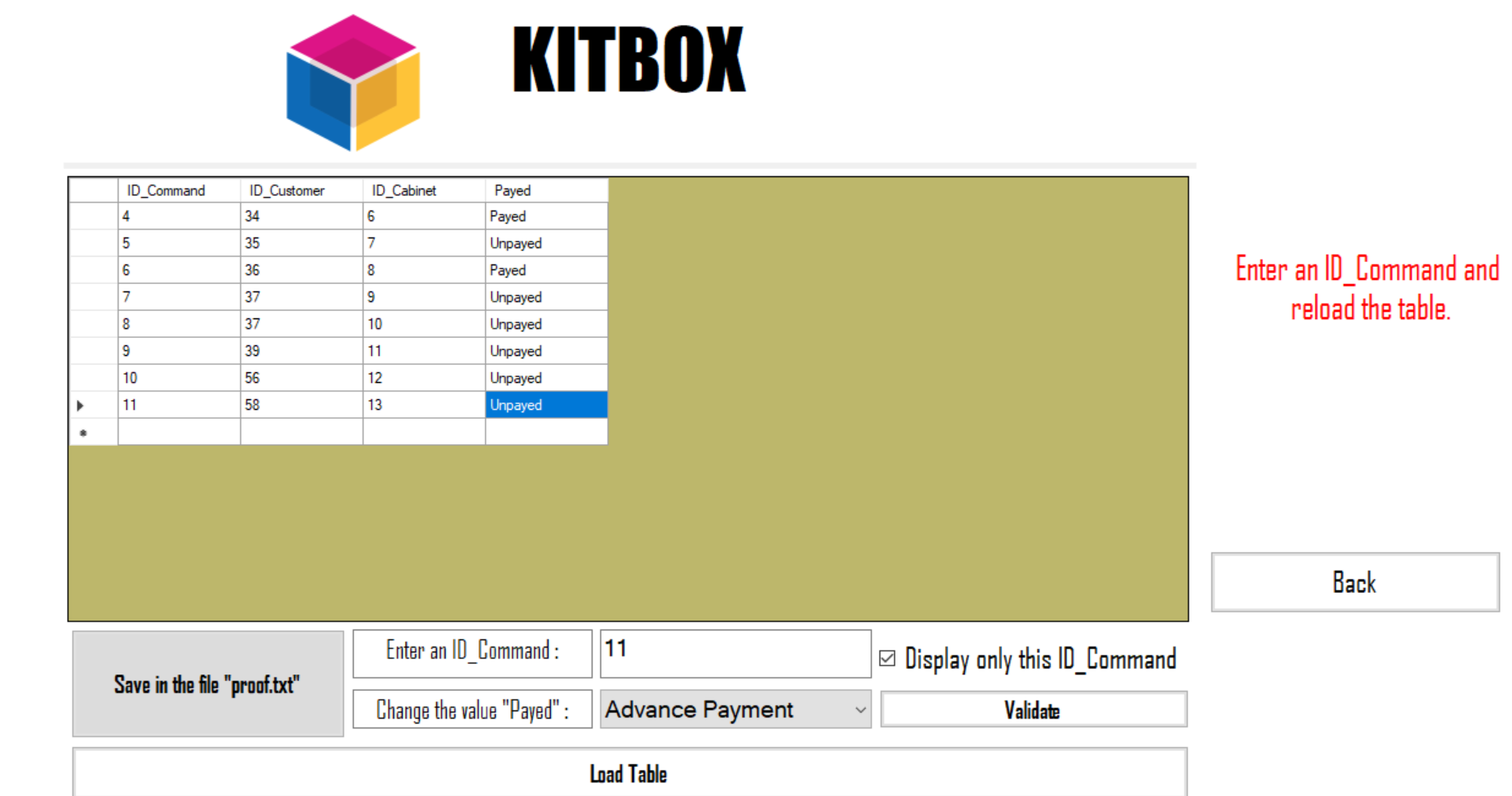
Lorsque l’utilisateur ferme le récapitulatif à l’aide du bouton **« Close Summary »,** une fenêtre l’avertira que sa commande a été sauvegardée l’interface sera remise à zéro. Il peut alors soit être d’accord et appuyer sur **« Ok »** soit sur **« Cancel »** et ainsi resté sur le résumé de la commande.



S’il est d’accord, l’utilisateur revient sur la page d’accueil.

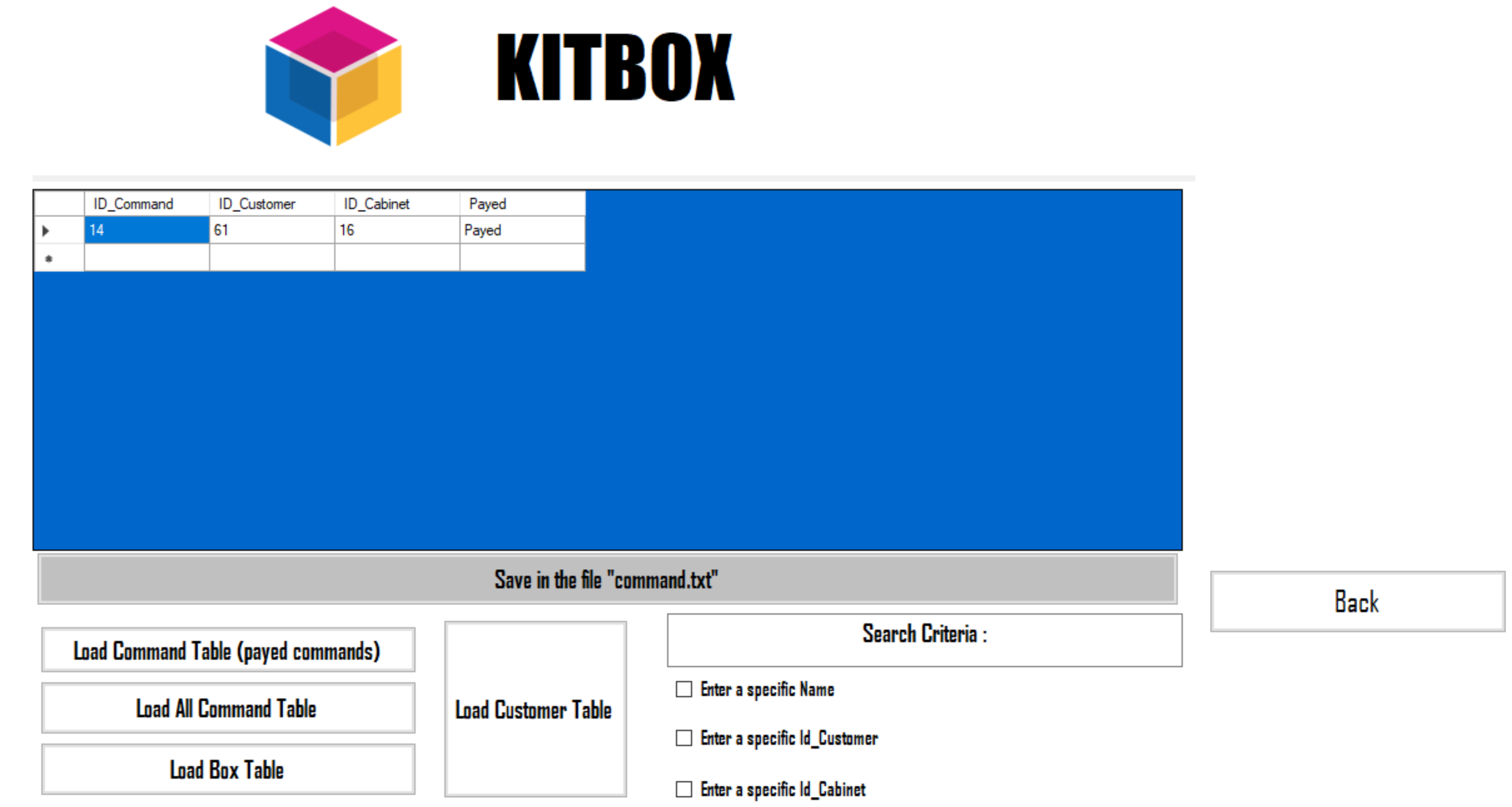
## Interface vendeur (Seller)

Après avoir entré le bon mot de passe, le vendeur peut modifier le statut d’une commande non payée (Unpayed) à celui de payée (Payed) ou bien celui d’avance si le client a déjà payé une partie de la commande (Advance Payment) via l’ID (identifiant) de la commande. S’il n’indique aucun ID, un message d’erreur apparait.



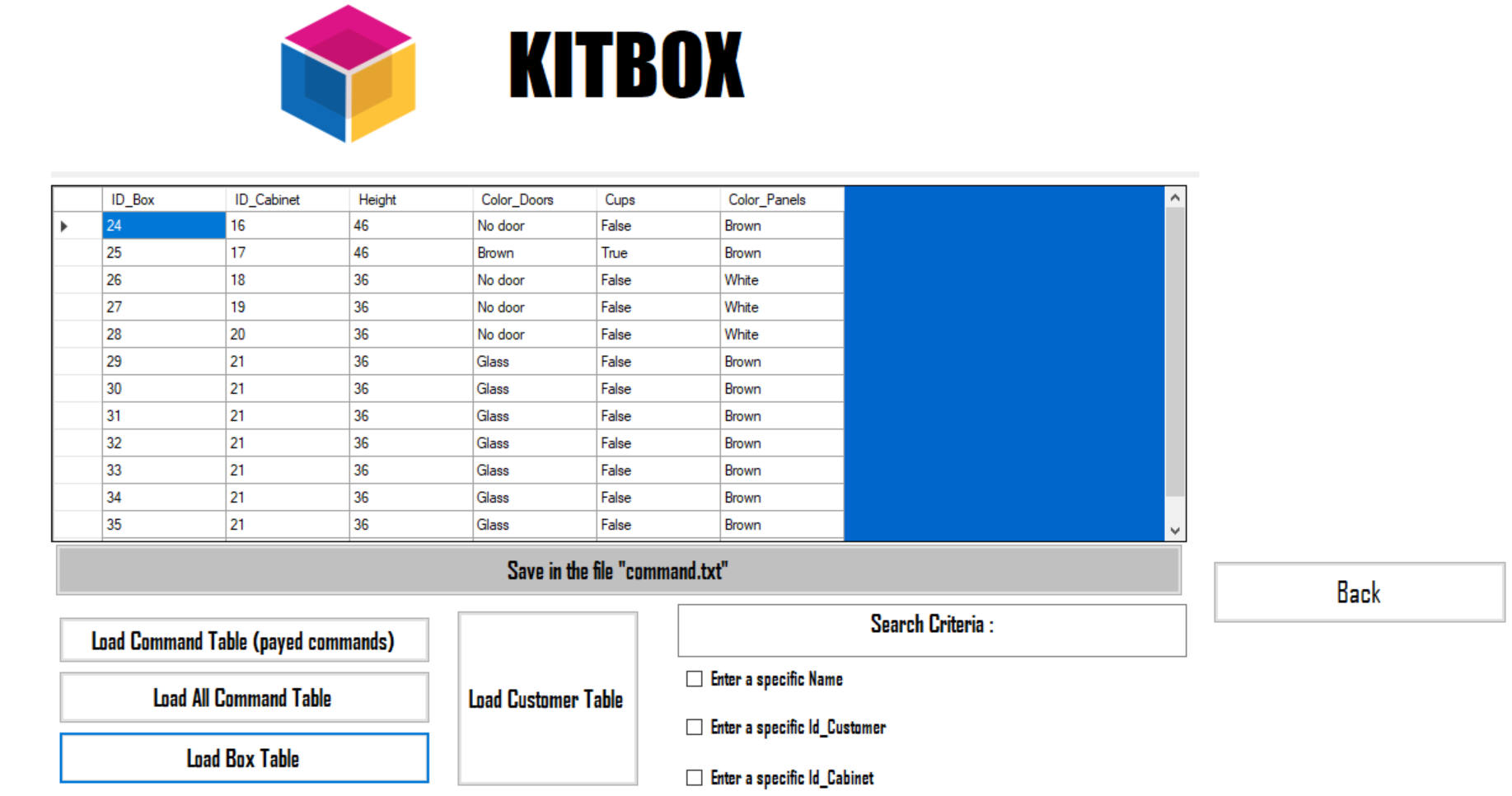
Le vendeur peut n’afficher que la commande qui l’intéresse en cochant l’option **« Display only this ID\_Command »** et en cliquant sur le bouton **« Load Table »**. Il peut changer le statut de la commande en cliquant sur le bouton « **Validate »** et enregistrer la commande dans un fichier texte qu’il pourra imprimer par la suite à l’aide du bouton **« Save in the file ‘proof.txt ‘ »**.  
Pour sortir de l’interface et revenir sur la page d’accueil, il suffira d’appuyer sur le bouton **« Back »**.

## Interface magasinier (Admin)



Il y a 4 boutons dont permettant d’afficher 4 tables. Le bouton **« Load Command Table (payed commands) »** permet d’afficher les commandes payées de la table commandes. Le magasinier pourra ainsi rassembler les pièces nécessaires à l’assemblage de l’armoire.

Le bouton **« Load All Command Table »** quant à lui permet d’afficher toutes les lignes de la table commandes, payées en tout ou en partie ou impayées. Le bouton **« Load Box Table »** permet de charger les lignes de la table contenant les différents casiers créés.



Le bouton **« Load Customer Table »** permet d’afficher la table reprenant tous les clients et leurs informations.  
Il est également possible une information selon 3 critères : **« Enter a specific Name »** qui permet d’afficher les commandes enregistrées sous un nom particulier.  
**« Enter a specifi Id\_Customer »** permet de retrouver les informations d’un client via son ID\_Customer.  
**« Enter a specific ID\_Cabinet »** permet d’afficher les différents casiers d’une armoire lorsqu’on charge la table Box Table.  
Enfin, le bouton **« Save in the file ‘command.txt’** » permet de sauvegarder ce qu’on a chargé avec une recherche selon un critère particulier ou non, dans un fichier texte qui sera imprimable par la suite.

On a 4 boutons pour 3 tables. Un bouton qui charge la table Box, l'autre c'est la table Customer, et le 3e c'est toute la table commande. Le Load Command Table (payed commands) sert à afficher que les lignes où c'est déjà payé. Ca permet à l'admin (qui est chargé de rassembler les pieces) de savoir direct ce qui est payé et ce qui ne l'est pas, si c'est payé, il rassemble les pièces

# Architecture

# Conclusion