|  |
| --- |
| Omar Egal Ahmed  30/05/2024 |

|  |
| --- |
| Rapport P\_UX |



Table des matières

[Introduction 0](#_Toc167996918)

[Interaction utilisateur 1](#_Toc167996919)

[Langue et internationalisation 1](#_Toc167996920)

[Navigation et flux de travail 1](#_Toc167996921)

[Indication de la langue en cours 1](#_Toc167996922)

[Adaptabilité de l'affichage 1](#_Toc167996923)

[Optimisation de l'espace 1](#_Toc167996924)

[Disposition physique 2](#_Toc167996925)

[2.Maquette des différents écrans de l’application 2](#_Toc167996926)

[3. Réalisation Architecture MVC 3](#_Toc167996927)

[Le Modèle 3](#_Toc167996928)

[Vue View 4](#_Toc167996929)

[Vue ViewselectNavigoorNot 5](#_Toc167996930)

[Vue ViewselectSpecialorNormaltickets 6](#_Toc167996931)

[Vue ViewselectSpecialtickets 7](#_Toc167996932)

[Vue ViewspecialTicketchoices 7](#_Toc167996933)

[Vue viewNormalticketChoices 8](#_Toc167996934)

[Vue "ViewmyAllchoices" 9](#_Toc167996935)

[Vue "ViewselectPaymentmethod" 9](#_Toc167996936)

[Vue ViewbillOrnot 10](#_Toc167996937)

[Contrôleur (modèle MVC) 11](#_Toc167996938)

[Conclusion 13](#_Toc167996939)

## Introduction

J'ai été missionné par la RATP pour proposer une nouvelle interface graphique pour les automates du métro parisien. Actuellement, les automates qui permettent d'acheter des billets ont un formulaire relativement simple, avec une molette de navigation, permettant de se déplacer vers le haut/bas, et un bouton pour valider son choix.

Le remplacement de ces machines a nécessité la création d’une nouvelle interface plus moderne et intuitive pour améliorer l’expérience utilisateur. L'interface doit non seulement simplifier la navigation et la sélection des billets, mais également intégrer des fonctionnalités supplémentaires telles que la gestion multilingue, de multiples options de paiement et une sélection de billets plus complexe, notamment des billets à prix réduit et des pass spéciaux pour des destinations comme Disneyland Paris ou des aéroports.

Dans ce cas, la mise en œuvre de l’architecture Model-View-Controller (MVC) devient cruciale. Il permet de séparer les préoccupations entre la gestion des données, la logique de l'application et la présentation visuelle. Ce chapitre détaille les différentes fonctions du contrôleur dans cette architecture, en se concentrant sur la manière dont il gère l'interaction entre la logique métier et l'interface utilisateur. Vous découvrirez comment les méthodes de contrôleur facilitent la navigation entre les vues, gèrent les tickets, intègrent la localisation multilingue et rendent l'interface réactive aux actions de l'utilisateur.Analyse de l’interface utilisateur existant

L'analyse de l'interface actuelle révèle plusieurs aspects nécessitant une attention particulière. Cette section présente une critique objective de l'interface existante ainsi que des propositions d'amélioration envisagées.

### Interaction utilisateur

L'interaction avec l'interface actuelle, caractérisée par l'utilisation de deux boutons pour valider ou annuler une action, ainsi qu'un rouleau pour défilement des menus, est critiquée. Ces méthodes d'interaction peuvent être considérées comme peu intuitives et nécessitant une amélioration pour une meilleure expérience utilisateur.

### Langue et internationalisation

Un défaut notable réside dans la présentation initiale des instructions en français, avant même que l'utilisateur ait eu l'opportunité de choisir sa langue préférée. L'absence d'une option pour l'anglais, langue internationale largement répandue, constitue également une lacune. Une solution à ce problème serait d'inclure dès le départ une option permettant à l'utilisateur de sélectionner sa langue de préférence.

### Navigation et flux de travail

Lors du premier choix entre le rechargement du Pass Navigo ou l'achat de billets, le défilement commence en plein milieu de l'écran, ce qui peut perturber le flux de l'utilisateur. Une amélioration potentielle serait de revoir la disposition des éléments de l'interface pour garantir un défilement fluide et intuitif.

### Indication de la langue en cours

Bien que toutes les langues soient affichées en bas de l'écran, il manque une indication claire de la langue actuellement utilisée. Pour une meilleure compréhension et facilité d'utilisation, il est recommandé d'ajouter un indicateur de langue en cours sur l'interface.

### Adaptabilité de l'affichage

Un autre point d'amélioration concerne l'adaptabilité de l'affichage en fonction du nombre d'éléments à l'écran. L'interface actuelle ne semble pas ajuster dynamiquement son agencement en fonction des contenus affichés, ce qui peut entraîner une présentation désordonnée et peu conviviale.

### Optimisation de l'espace

La bande de gauche de l'interface est principalement dédiée à l'affichage de la réservation en cours, mais son utilisation semble inefficace, car elle ne prend qu'un tiers de la largeur de la bande. Une révision de la répartition de l'espace pourrait permettre une utilisation plus efficace de cette zone.

### Disposition physique

Enfin, des problèmes ergonomiques sont observés, tels que la position trop basse du support pour les sacs et la disposition peu pratique de la fente pour insérer les pièces. Ces éléments physiques de l'interface nécessitent une révision pour garantir un confort d'utilisation optimal pour les utilisateurs.

## 2.Maquette des différents écrans de l’application

Une image contenant texte, diagramme, écriture manuscrite, Plan

Description générée automatiquement

Figure 1. Maquette des vues 1-2 et 7-9.

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

Figure 1. Maquette des vues 1-2 et 7-9.

## 3. Réalisation Architecture MVC

### Le Modèle

#### Définition des Propriétés de Base

Le modèle contient plusieurs attributs pour représenter les prix des différents types de billets, ainsi que leurs noms et le solde actuel de l'utilisateur.

#### Explication des Propriétés

Prix des billets : Les propriétés \_priceStandardticket, \_priceReducedticket, \_pricesChessyDisneyticketOne, \_pricesAirportyticketOne, et \_pricesParisVisiteticketOne stockent les prix des différents types de billets.

Noms des billets : Les propriétés \_nameChessyDisneyticket, \_nameAirportyticket, \_nameParisVisiteticket, \_nameReducedticket, et \_nameStandardticket stockent les noms des différents types de billets.

Solde actuel : La propriété Balance stocke le solde actuel de l'utilisateur.

Liste des billets : La propriété Tickets est une liste d'objets Ticket représentant les billets achetés.

Référence au contrôleur : La propriété Controller stocke une référence au contrôleur de l'application, permettant au modèle d'interagir avec la logique de contrôle.

#### Constructeur

Le constructeur de la classe Model est utilisé pour initialiser les propriétés par défaut. Actuellement, Il est vide mais peut être étendu pour contenir une initialisation spécifique si nécessaire.

Ce modèle constitue la base de la gestion des données dans notre application, fournissant un moyen structuré de stocker et de manipuler les informations sur les tickets et les transactions. L'étape suivante consiste à intégrer ce modèle avec des vues et des contrôleurs pour créer une application fonctionnelle qui suit le modèle MVC.

### Vue View

#### Description de la Vue

Cette première vue est l'écran d'accueil de l'application de vente de billets, où les utilisateurs peuvent sélectionner leur langue préférée parmi cinq options : français, anglais, allemand, espagnol et italien. Cette vue est cruciale pour une bonne expérience utilisateur (UX), car elle permet aux utilisateurs de naviguer facilement dans leur langue maternelle ou une langue avec laquelle ils sont à l'aise.

Cette vue contient les composant suivant :

* Labels : Indiquent aux utilisateurs de sélectionner une langue.
* Panel : Contient les boutons de sélection de langue.
* Boutons : Permettent aux utilisateurs de choisir leur langue préférée.

Voici un résumé des principaux éléments :

* Label lblSelectlangages : Affiche le texte "Sélectionner une langue".
* Panel pnlSelectlanguages : Contient les boutons de sélection de langue.
* Boutons de langue : Chaque bouton (par exemple, btnItalianinFirstpage, btnDeutshinFirstpage, btnSpanishinFirstpage) est configuré pour une langue spécifique, avec des labels correspondants pour améliorer la clarté.

La propriété Controller est définie pour gérer les interactions entre la vue et le contrôleur est l’est présentes dans toutes mes vues et ne sera plus spécifier dans le reste du chapitre réalisation afin de fluidifier votre lecture.

Méthode UpdateLang(ResourceManager \_resourceManager) : Cette méthode itère à travers tous les contrôles de la vue pour mettre à jour les textes selon la langue sélectionnée, en utilisant un ResourceManager.

#### UX et Choix de Conception

* Simplicité et Clarté : L'écran de sélection de la langue est simple et direct, permettant aux utilisateurs de comprendre facilement ce qu'ils doivent faire dès qu'ils arrivent sur l'écran.
* Accessibilité : En proposant plusieurs langues, l'application devient accessible à une large audience internationale.
* Feedback Immédiat : Les boutons réagissent immédiatement au clic, ce qui donne un retour instantané à l'utilisateur, améliorant ainsi l'expérience utilisateur.

#### Méthodes évènementielles

Chaque bouton de langue a un gestionnaire d'événements associé qui déclenche l'action correspondante :

1. btnEnglishinFirstpage\_Click : Change la langue à l'anglais et passe à l'écran suivant.
2. btnDeutshinFirstpage\_Click, btnSpanishinFirstpage\_Click, et btnItalianinFirstpage\_Click : Chaque bouton appelle la méthode Controller.ShowViewselectNavigoOrnot() pour passer à l'écran suivant après la sélection de la langue.

#### Conclusion

Cette première vue constitue une partie essentielle de l'application, assurant que les utilisateurs puissent commencer leur expérience dans une langue familière. Le design et la logique derrière cette vue sont conçus pour être intuitifs et inclusifs, respectant les principes fondamentaux de l'ergonomie et de l'expérience utilisateur.

Cette première vue a été analysé et présenté en détails pour donner une vision précise de comment a été réaliser l’application. Les vues se ressemblant à peu près toutes, les sous-chapitre suivant se concentreront sur l’ergonomie UX apporter et si nécessaire détaillera les éléments spécifiques complexe à chaque vue.

### Vue ViewselectNavigoorNot

#### Description de la Vue

Cette vue offre à l'utilisateur la possibilité de choisir s'il possède ou non une carte Navigo. Elle se présente avec une option pour indiquer qu'il possède déjà une carte et une autre pour signaler qu'il n'en possède pas. Le bouton de retour permet à l'utilisateur de revenir en arrière s'il souhaite apporter des modifications à sa sélection.

#### Explication de l'agencement et de l'UX :

##### Bouton de retour

Placé en haut à gauche de la fenêtre, ce bouton assure une navigation fluide et intuitive. Il permet à l'utilisateur de revenir en arrière à tout moment, ce qui renforce la facilité d'utilisation de l'application.

##### Options de sélection

Les deux options sont clairement présentées dans le panneau principal de la vue. L'agencement clair et simple permet à l'utilisateur de faire un choix rapidement et efficacement, sans confusion.

##### Feedback visuel

Les boutons sont conçus de manière à fournir un feedback visuel immédiat lorsqu'ils sont survolés ou cliqués, garantissant ainsi une expérience utilisateur agréable et réactive.

##### Texte explicatif

Le texte des boutons est précis et explicite, ce qui aide l'utilisateur à comprendre rapidement les options disponibles. Cela réduit les risques d'erreurs et de confusion.

Cette disposition ergonomique vise à optimiser l'expérience utilisateur en offrant une interface simple, intuitive et efficace pour la sélection de l'état de possession d'une carte Navigo.

#### Internationalisation (I18N) :

L’I18N est implémenter de la même manière dans les vues de mon application. Voici comment elle est gérée :

### Vue ViewselectSpecialorNormaltickets

#### Description de la Vue

La vue est structurée en plusieurs panneaux (Panel), chacun avec un objectif spécifique, comme l'en-tête, le pied de page, les informations actuelles sur l'achat et les détails des billets spéciaux.

Les contrôles de l'interface utilisateur, tels que les boutons (Button), les étiquettes (Label), sont placés dans les panneaux selon leur fonctionnalité.

#### Expérience Utilisateur (UX)

##### Clarté et Hiérarchie de l'Information

La vue est organisée de manière que l'utilisateur comprenne immédiatement les options disponibles. Le texte "Faites un choix" dans le header guide l'utilisateur sur le but de la vue.

##### Facilité de Navigation

Les boutons clairs et bien étiquetés facilitent la navigation entre les différentes sections de l'application. Le bouton "RETOUR" dans le header permet à l'utilisateur de revenir en arrière facilement.

##### Réactivité

Les boutons réagissent immédiatement au clic de l'utilisateur, offrant ainsi une expérience fluide et réactive.

#### Internationalisation (I18N) :

L’I18N est implémenter de la même manière dans les vues de mon application. Voici comment elle est gérée :

##### Support Multilingue

Les boutons dans le footer permettent à l'utilisateur de sélectionner la langue de l'interface. Cela montre une prise en charge efficace de l'internationalisation, permettant à l'application d'être utilisée dans différentes langues.

##### ResourceManager

L'utilisation de ResourceManager dans la méthode UpdateLang facilite la gestion des ressources de localisation, permettant une traduction efficace du texte de l'interface utilisateur.

En combinant une conception axée sur l'UX avec une approche efficace de l'internationalisation, cette vue vise à offrir une expérience utilisateur agréable et accessible à un public diversifié, indépendamment de leur langue préférée.

### Vue ViewselectSpecialtickets

#### Description de la Vue

La vue est structurée en plusieurs panneaux (Panel), chacun avec un objectif spécifique, comme l'en-tête, le pied de page, les informations actuelles sur l'achat et les détails des billets spéciaux.

Les contrôles de l'interface utilisateur, tels que les boutons (Button), les étiquettes (Label), sont placés dans les panneaux selon leur fonctionnalité.

#### Expérience Utilisateur (UX)

##### Organisation de l'interface utilisateur

La disposition des éléments tels que l'en-tête, le pied de page et les panneaux d'informations est conçue de manière que l'utilisateur puisse facilement comprendre où trouver les options de tickets spéciaux et comment naviguer dans l'application.

##### Clarté des actions

Les boutons de choix de tickets spéciaux sont clairement libellés, ce qui permet à l'utilisateur de comprendre instantanément les options disponibles.

L'utilisation d'étiquettes numérotées dans le panneau d'informations facilite la correspondance entre les descriptions et les boutons de choix.

##### Navigation cohérente

Le bouton "RETOUR" dans l'en-tête assure une navigation cohérente et prévisible pour l'utilisateur, lui permettant de revenir en arrière si nécessaire.

##### Feedback utilisateur

Lorsque l'utilisateur sélectionne un type de ticket spécial, il reçoit un retour d'information immédiat sous la forme de la navigation vers la prochaine vue correspondant à son choix.

#### Internationalisation (I18N)

L’I18N est implémenter de la même façon que dans les autres dans cette vue.

### Vue ViewspecialTicketchoices

#### Description de la Vue

La vue est structurée en plusieurs panneaux (Panel), chacun avec un objectif spécifique, comme l'en-tête, le pied de page, les informations actuelles sur l'achat et les détails des billets spéciaux.

Les contrôles de l'interface utilisateur, tels que les boutons (Button), les étiquettes (Label), les sélecteurs de date (DateTimePicker) et les boutons radio (RadioButton), sont placés dans les panneaux selon leur fonctionnalité.

#### Interaction Utilisateur (UX)

Les boutons sont associés à des événements de clic pour effectuer des actions spécifiques, comme retourner à la vue précédente ou valider les informations entrées.

Les événements Shown, Load, et Activated sont utilisés pour mettre à jour dynamiquement le texte de l'en-tête en fonction des données actuelles.

##### Gestion des Événements

Les événements de clic des boutons sont liés à des méthodes dans le contrôleur (Controller), ce qui délègue la logique métier à une autre couche de l'application.

Par exemple, le clic sur le bouton "VALIDER" (btnValidatorInfos\_Click) déclenche plusieurs actions liées à la validation des informations saisies et à l'affichage de messages appropriés.

#### Internationalisation (I18N)

L’I18N est implémenter de la même façon que dans les autres dans cette vue.

### Vue viewNormalticketChoices

#### Description de la Vue

La vue est structurée en plusieurs panneaux (Panel), chacun avec un objectif spécifique, comme l'en-tête, le pied de page, les informations actuelles sur l'achat et les détails des billets normaux.

Les contrôles de l'interface utilisateur, tels que les boutons (Button), les étiquettes (Label), les sélecteurs de date (DateTimePicker) et les boutons «plus » et « Moins » sont placés dans les panneaux selon leur fonctionnalité.

#### Interactivité et UX

Les boutons permettant d'ajuster les quantités de tickets standard et réduits offrent une expérience utilisateur conviviale et interactive. Ils permettent aux utilisateurs de sélectionner facilement les quantités souhaitées, améliorant ainsi l'ergonomie de l'application.

#### Gestion des événements

Les événements associés aux boutons sont correctement gérés, assurant une interaction fluide avec l'interface. Les actions pour augmenter ou diminuer le nombre de tickets, ainsi que la validation des informations saisies, sont effectuées de manière efficace et cohérente.

#### Internationalisation (I18N)

L’I18N est implémenter de la même façon que dans les autres dans cette vue.

### Vue "ViewmyAllchoices"

#### Description de la Vue

La vue "viewmyAlchoices" constitue une composante cruciale de l'expérience utilisateur dans l'application P\_UX\_ACD\_EgalAhmeOmar. Cette vue de l’application présente une structure similaire aux autres vues et dans l'ensemble, mais elle a toute de même la particularité de contenir le panier d’achat des tickets.

##### Interactivité et UX

Cette vue de l’application présente une structure similaire aux autres vues et dans l'ensemble, cette conception met l'accent sur la clarté et la facilité d'utilisation, en fournissant à l'utilisateur des repères visuels et des options de navigation intuitives pour une expérience agréable et fluide.

##### Séparation des préoccupations

Le code de la vue suit le principe de séparation des préoccupations en déléguant la logique métier à un contrôleur externe. Cela permet de maintenir un code modulaire et réutilisable, favorisant ainsi la scalabilité et la facilité de maintenance de l'application.

##### Gestion des événements

La vue répond aux interactions de l'utilisateur à l'aide de gestionnaires d'événements appropriés. Les méthodes associées aux boutons "RETOUR", "Prochaine Achat" et "PAYER" sont des exemples de cette gestion des événements, garantissant une interaction fluide et intuitive avec l'application.

##### Activation de la vue

L'événement Activated est utilisé pour exécuter des actions spécifiques lorsque la vue est activée. Cela permet d'effectuer des opérations contextuelles, telles que l'affichage dynamique de contenu, chaque fois que la vue devient active.

#### Internationalisation (I18n)

L’I18N est implémenter de la même façon que dans les autres dans cette vue.

### Vue "ViewselectPaymentmethod"

#### Description de la Vue

La vue "ViewselectPaymentmethod" constitue une composante cruciale de l'expérience utilisateur dans l'application P\_UX\_ACD\_EgalAhmeOmar. Cette vue de l’application présente une structure similaire aux autres vues et dans l'ensemble, elle a de différents de autres vue que elle gère les option de paiement de l’utilisateur.

#### Interactivité et UX

##### Gestion de l'En-tête

L'en-tête de la vue comprend un libellé indiquant le montant total à payer, offrant ainsi à l'utilisateur une indication claire sur le contexte de la page.

##### Navigation Intuitive

Le bouton "RETOUR" dans l'en-tête permet à l'utilisateur de revenir en arrière dans le flux de navigation de l'application, garantissant ainsi une expérience de navigation fluide et intuitive.

##### Sélection de Mode de Paiement

La vue propose plusieurs options de paiement, telles que l'utilisation de Google Pay, le paiement par cartes bancaires, ou encore l'utilisation de pièces et billets. Chaque option est accompagnée d'un bouton de validation, offrant à l'utilisateur un moyen clair et intuitif de sélectionner son mode de paiement préféré.

##### Affichage Dynamique

La vue affiche également des informations dynamiques sur le mode de paiement actuellement sélectionné, garantissant ainsi à l'utilisateur une rétroaction en temps réel sur ses choix.

##### Internationalisation (I18n)

L’I18N est implémenter de la même façon que dans les autres dans cette vue.

### Vue ViewbillOrnot

#### Description de la Vue

Cette vue est l’étape suivante dans le processus d'achat de billets où l'utilisateur est invité à confirmer s'il souhaite recevoir un reçu pour son achat. Cette vue est importante pour la gestion des transactions et la satisfaction des clients, leur permettant de recevoir une preuve d'achat s'ils le souhaitent.

#### Composants de de la vue

* Panel d'En-Tête : Contient une étiquette pour demander à l'utilisateur s'il souhaite un reçu.
* Panel de Pied de Page : Contient des boutons pour changer la langue de l'interface et un bouton pour arrêter le processus d'achat.
* Panel d'Achat Actuel : Affiche les détails de l'achat actuel.
* Panel d'Informations : Contient des boutons pour la confirmation du reçu.
* Label lbltextinHeaderinBillorNot : Affiche la question "Voulez-vous une quittance ?".
* Boutons de langue dans le pied de page : Permettent à l'utilisateur de changer la langue de l'interface.
* Boutons btnYesforReceipt et btnNoforReceipt : Permettent à l'utilisateur de répondre à la question posée et de confirmer ou non s'il souhaite un reçu.

#### UX et Choix de Conception

##### Simplicité et Clarté

La vue est simple et claire, posant une question directe à l'utilisateur avec des options de réponse faciles à comprendre.

##### Feedback Immédiat

Les boutons de confirmation réagissent immédiatement au clic, ce qui donne un retour instantané à l'utilisateur.

##### Accessibilité

Les options de changement de langue dans le pied de page rendent l'application accessible à une audience internationale.

##### Cohérence

La mise en page est cohérente avec d'autres vues, assurant une expérience utilisateur fluide.

#### Conclusion

Cette dernière vue constitue une partie essentielle du processus d'achat, assurant que les utilisateurs peuvent confirmer s'ils souhaitent un reçu. Le design et la logique derrière cette vue sont conçus pour être intuitifs, respectant les principes fondamentaux de l'ergonomie et de l'expérience utilisateur.

## Contrôleur (modèle MVC)

Les contrôleurs sont un composant important de l'architecture Model-View-Controller (MVC). Il sert d'intermédiaire entre le modèle, qui contient la logique métier, et la vue, qui gère l'interface utilisateur. Le rôle principal d'un contrôleur est de répondre aux actions de l'utilisateur, de manipuler le modèle et de mettre à jour la vue en conséquence. Dans cette section, nous explorerons les fonctions clés des contrôleurs dans les applications de billetterie.

#### Gestion des Vues

Les contrôleurs initialisent et gèrent les différentes vues de l'application. Il dispose de méthodes pour afficher ou masquer des formulaires spécifiques selon les besoins. Par exemple, des méthodes telles que ShowViewselectNavigoOrnot, ShowViewselectSpecialtickets et ShowViewSpecialticketChoices vous permettent de naviguer entre différentes vues en fonction des actions de l'utilisateur.

#### Gestion des Tickets Spéciaux

Les billets spéciaux tels que les billets Chessy Disney, les billets d'aéroport et les billets Paris Visite sont gérés par le contrôleur. Le nom de ces tickets est assurée grâce à l'utilisation d'un gestionnaire de ressources (ResourceManager) pour les récupérer dans différentes langues. On recourt à des méthodes telles que GetlabelHeardertextChessydisneyTicket et GetnewSpecialticket afin d'obtenir les données des billets et de générer de nouveaux billets en fonction des critères choisis par l'utilisateur.

#### Sélection et Validation des Billets

Le contrôleur offre des méthodes pour sélectionner et valider les billets. Par exemple, GetnewNormalticket et GetnewReducedticket créent de nouveaux billets normaux et réduits respectivement. La méthode ChecktwoLabelcontainZeroTicket vérifie si aucune sélection de billets n'a été faite et affiche un message d'erreur le cas échéant.

#### Gestion des Langues

La gestion des langues constitue un élément essentiel du contrôleur. En utilisant la méthode Lang, le contrôleur a la possibilité de modifier la langue de l'application. Cela met à jour le ResourceManager avec la nouvelle langue et met à jour toutes les vues pour refléter cette langue. Cela favorise une expérience utilisateur cohérente.

#### Mise à Jour de l'Interface Utilisateur

Le contrôleur propose des méthodes permettant de mettre à jour de manière dynamique l'interface utilisateur. Par exemple, DisplayTickets présente sur un panneau les informations sur les billets achetés, en regroupant les billets par catégorie et en indiquant le nombre total de billets et le prix total. Pour ajouter des labels de des nombre de billets et des prix au panneau d'affichage, on utilise des méthodes auxiliaires telles qu'AddLabelToPanel, AddLabelCountToPanel et AddLabelPriceToPanel.

#### Messages et Notifications

Afin d'optimiser la communication avec les utilisateurs, le contrôleur affiche des messages et des notifications en réponse à certaines actions. Par exemple, le message ShowmessageZeroTicket montre un message lorsque l'utilisateur tente de poursuivre sans choisir de billets, tandis que le message ShowvalidTicket affiche un message de confirmation une fois que les billets sont validés.

#### Conclusion

En bref, le rôle du contrôleur est essentiel pour orchestrer les interactions des utilisateurs, gérer les données et mettre à jour l'interface utilisateur. Il garantit le traitement adéquat des actions de l'utilisateur et la cohérence et l'information de l'état de l'application.

## Conclusion

En résumé, la conception et la mise en œuvre d'une nouvelle interface pour les distributeurs automatiques du métro parisien nécessitent une réflexion approfondie sur la conception et la mise en œuvre des fonctionnalités essentielles. Il s'est avéré crucial d'adopter une architecture MVC afin de garantir une répartition claire des tâches, ce qui se traduit par une amélioration de la durabilité et de l'évolutivité du système.

Il est primordial de recourir à des méthodes du contrôleur afin de simplifier la transition entre les différentes vues, gérer les divers types de tickets disponibles et intégrer des fonctionnalités interlangues. De plus, leur expérience utilisateur est agréable et intuitive, ce qui répond aux différentes exigences des utilisateurs du métro parisien.

Ce projet met en évidence l'importance d'une interface utilisateur bien élaborée afin d'améliorer non seulement l'efficacité des transactions, mais aussi de proposer un service accessible et convivial. La mise en place de cette nouvelle interface constitue une étape cruciale dans la modernisation du service RATP, favorisant ainsi l'amélioration de la satisfaction des utilisateurs et une gestion plus efficace des opérations de billets.