Résumé de Stage
BTS SIO - SLAM

Nom : Dylan Balmigere
Année scolaire : 2024 - 2025
Lycée : Jean Lurçat

Entreprise d'accueil : Euresto
Période du stage : Du 6 janvier au 14 février 2025

Introduction

Ce document présente un résumé des différentes missions effectuées lors de mon stage. Il met en avant les tâches réalisées, les compétences acquises ainsi que les problématiques rencontrées et les solutions apportées.

J'ai effectué mon stage de deuxième année au sein de l'entreprise Euresto, une société spécialisée dans la programmation informatique, située à Thuir, dans les Pyrénées-Orientales. Euresto conçoit, développe et diffuse des logiciels ainsi que des outils de gestion, tout en proposant des services de conseil, d'installation, de formation et de maintenance informatique. L'entreprise utilise principalement le langage de développement **WINDEV** pour la création de ses applications.

Mon stage s'est déroulé sur une durée de 6 semaines durant lesquelles j'ai pu m'investir pleinement dans différentes missions en lien avec le développement logiciel. J'ai été intégré à l'équipe de développement de l'entreprise Euresto, où j'ai participé à plusieurs projets d'envergure.

Ce stage m'a permis de renforcer mes compétences en développement informatique et de mieux comprendre les enjeux d'un environnement professionnel, notamment en matière de gestion de projet, de communication en équipe et d'adaptation aux besoins des utilisateurs.

Ce que j’ai fait sur le projet Windev Mobile (Niveau Visuel UI)

Je vais donc vous présenter toutes les choses que j’ai faites au niveau de Windev Mobile en commençant par ce que j’ai fait au niveau du visuel.

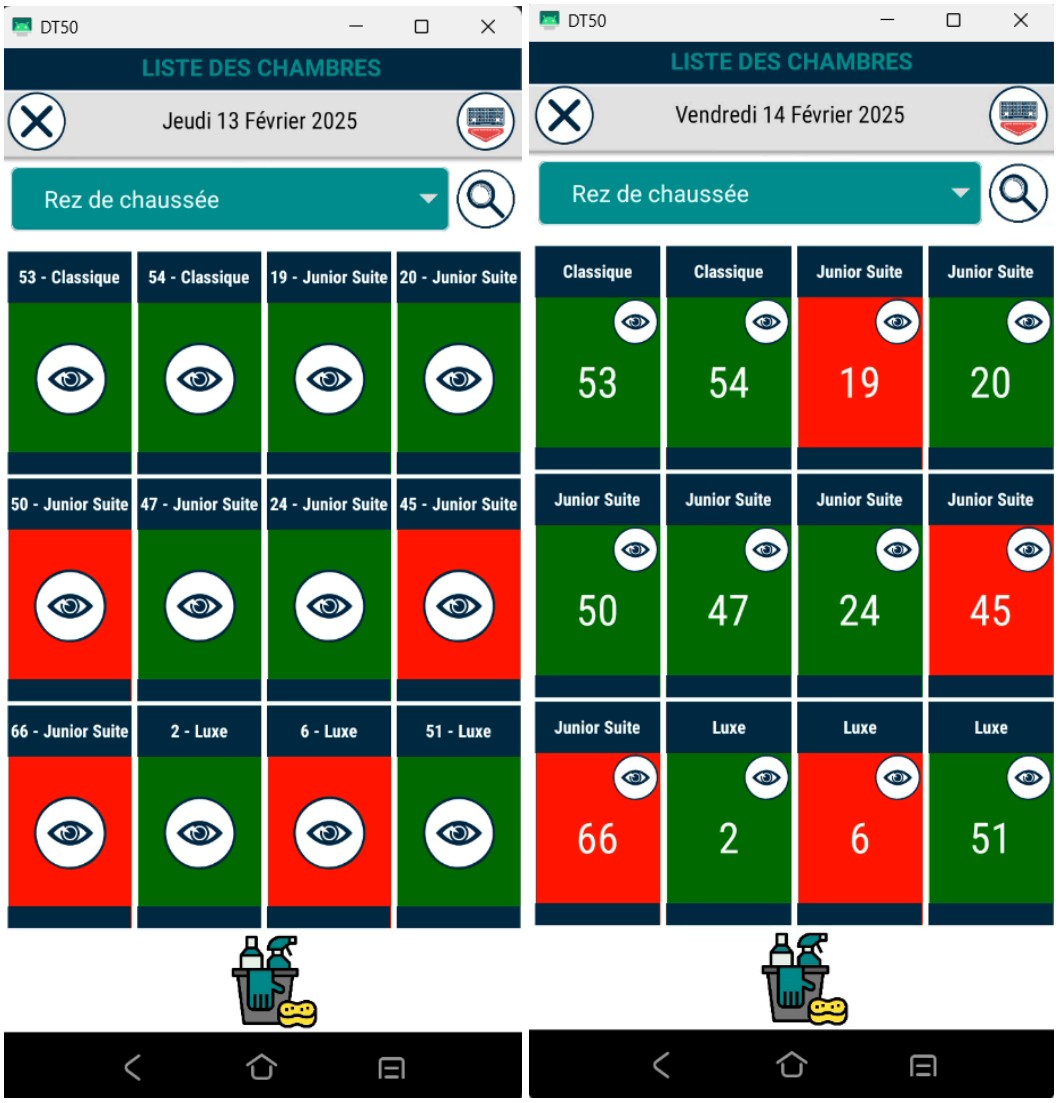
Ma tâche principale a consisté à créer une fenêtre affichant la liste des chambres de l’établissement connecté et qui permettra aux personnes ménagères de renseigner plusieurs informations sur la chambre sélectionné tel que l’état de la chambre (si elle est propre ou sale), ajouter un commentaire si un problème est rencontré et même pouvoir mettre une photo afin de bien comprendre la situation.
Une fois toutes les informations renseignées, il suffira juste d’appuyer sur un bouton de validation afin d’envoyer les données sur une base locale afin qu’elles y soit stockées.

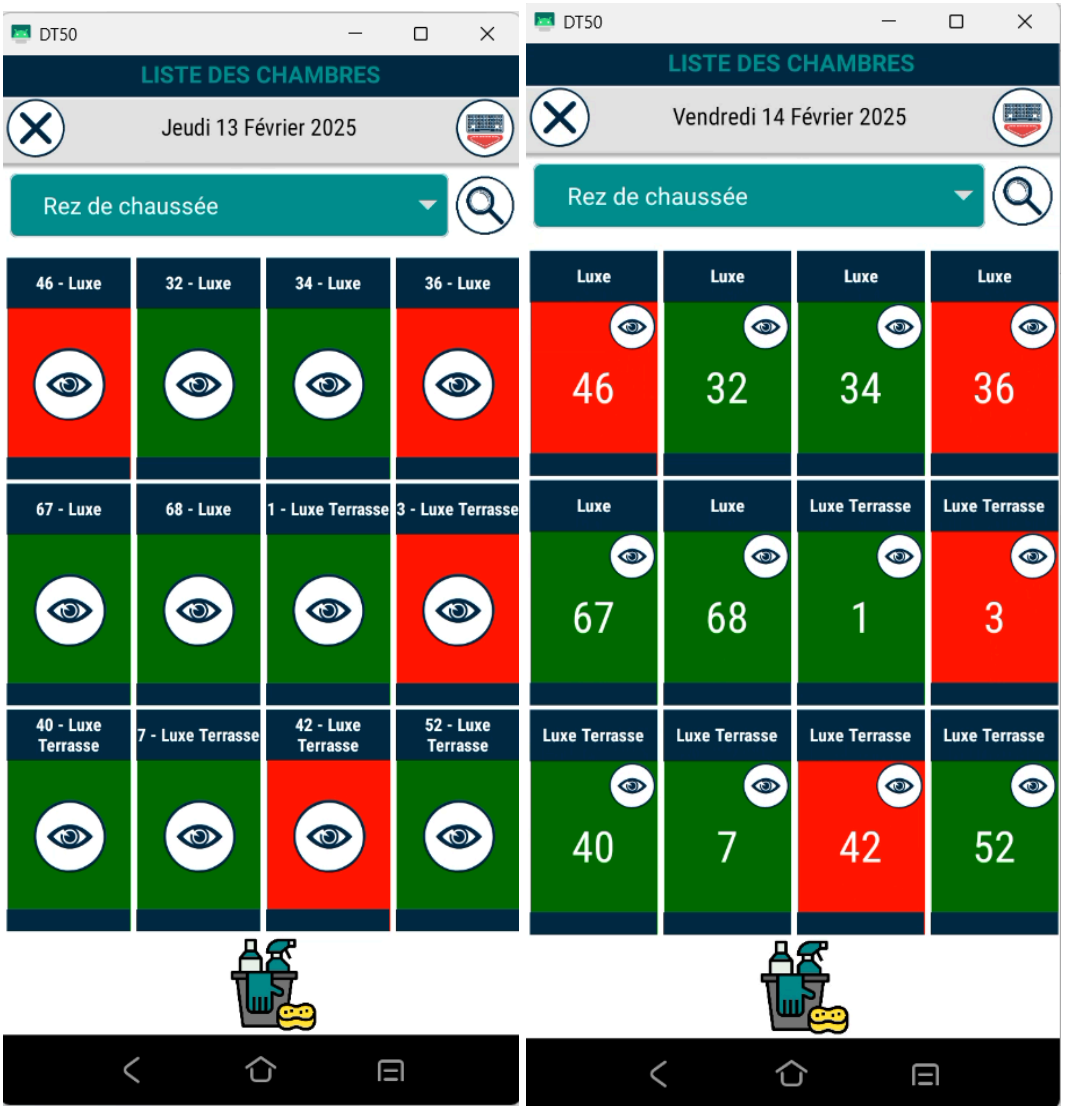
Premièrement, pour que je puisse afficher la fenêtre que j’ai créé, il a fallu que je rajoute un onglet dans le menu principal de l’application mobile qui sera uniquement accessible aux personnes connectés disposant des droits pour l’afficher et qui me permettra donc, justement, d’accéder à cette fenêtre comme je vais vous le montrer ci dessous :

Vous pouvez donc bien voir au centre de cette fenêtre le menu Entretien qui affiche un bouton “Gestion des chambres” qui permet d’afficher la fameuse fenêtre que j’ai développée et un autre bouton “Paramètres Entretien” qui permet de modifier diverses options au sein de la fenêtre que j’ai créé.

Avant d’afficher les éléments principaux de ma fenêtre, on va afficher le tout premier plan de la fenêtre qui est juste une fenêtre de chargement qui va s’afficher le temps de récupérer la liste complète des chambres :

Nous arrivons donc sur le second plan de la même fenêtre ou l'on affiche cette fois-ci la liste de toutes les chambres disponibles pour cet établissement comme vous pouvez le voir ci dessous (le design définitif est celui de droite) :





Sur cette fenêtre, on peut donc retrouver une multitude de choses telles que la date actuelle du jour, on a la liste de toutes les chambres qui est affiché et qui se retrouve dans une liste que l'on peut faire défiler comme montré ci dessus avec d'autres chambres de la liste.

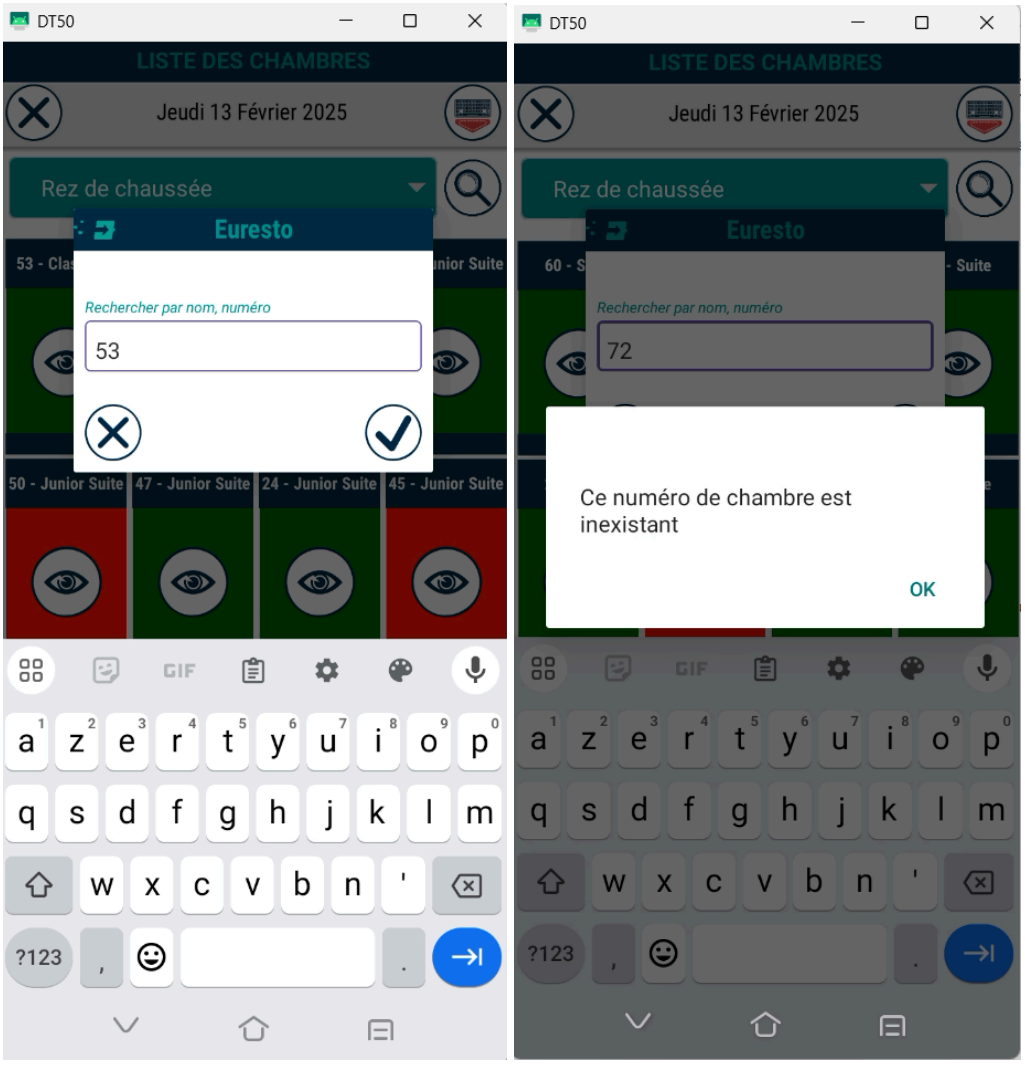
Pour chaque chambres, vous avez une couleur de fond qui dépendra de l'état de la chambre (Sale ou Propre), vous avez le numéro de la chambre avec la catégorie dans laquelle elle se trouve, il y a aussi un bouton en forme d'oeil (que l'on peut

modifier) qui permet, lorsqu'il est appuyé, d'afficher les informations de la chambre sélectionné.

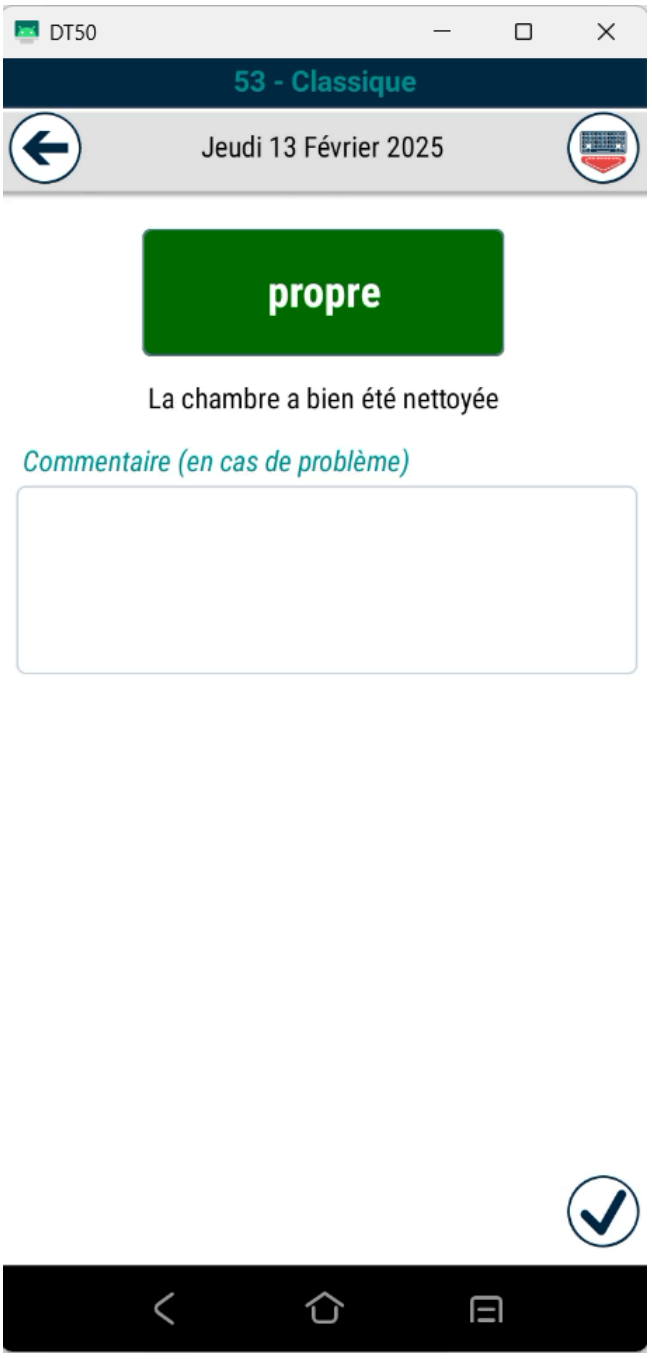
Vous avez un champ Combo qui permet de sélectionner un étage en particulier et ainsi de modifier la liste des chambres en affichant uniquement les chambres se trouvant à cet étage, je ne l'ai pas configuré en code car l'établissement sur lequel je suis connecté ne possède pas de chambres en étages, elles sont toutes au même niveau. Je vous montrerai plus loin la logique au niveau du code de ce sélecteur..

Dernière chose au niveau de cette fenêtre, vous avez un bouton en haut à droite qui permet, lorsqu'il est appuyé, d'ouvrir une petite fenêtre Popup qui s'ouvre en premier plan et sur laquelle vous pouvez entrer le numéro d'une chambre présente dans la liste et ainsi arriver directement sur ces détails lorsque le bouton de validation de la recherche est appuyé. Il y a un autre cas qui est le fait que si vous rentrez un numéro de chambre non présent dans la liste, alors une information va s'afficher vous disant que ce numéro de chambre est inexistant.

Je vous montre les 2 cas ci-dessous :

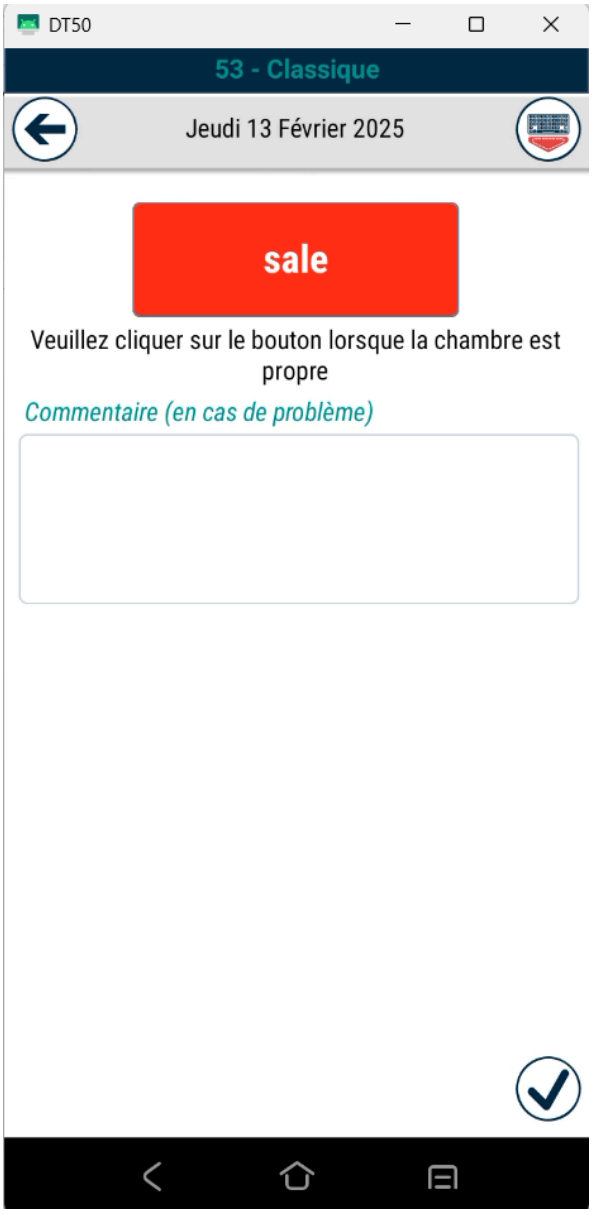


Si la recherche est validé ou que, directement dans la liste des chambres, on a appuyé sur le bouton oeil de l'une des chambres, alors on passe au troisième plan de cette même fenêtre qui permet d'afficher les diverses informations de celle-ci (Dans mon exemple, on va le faire avec la chambre n°53) :



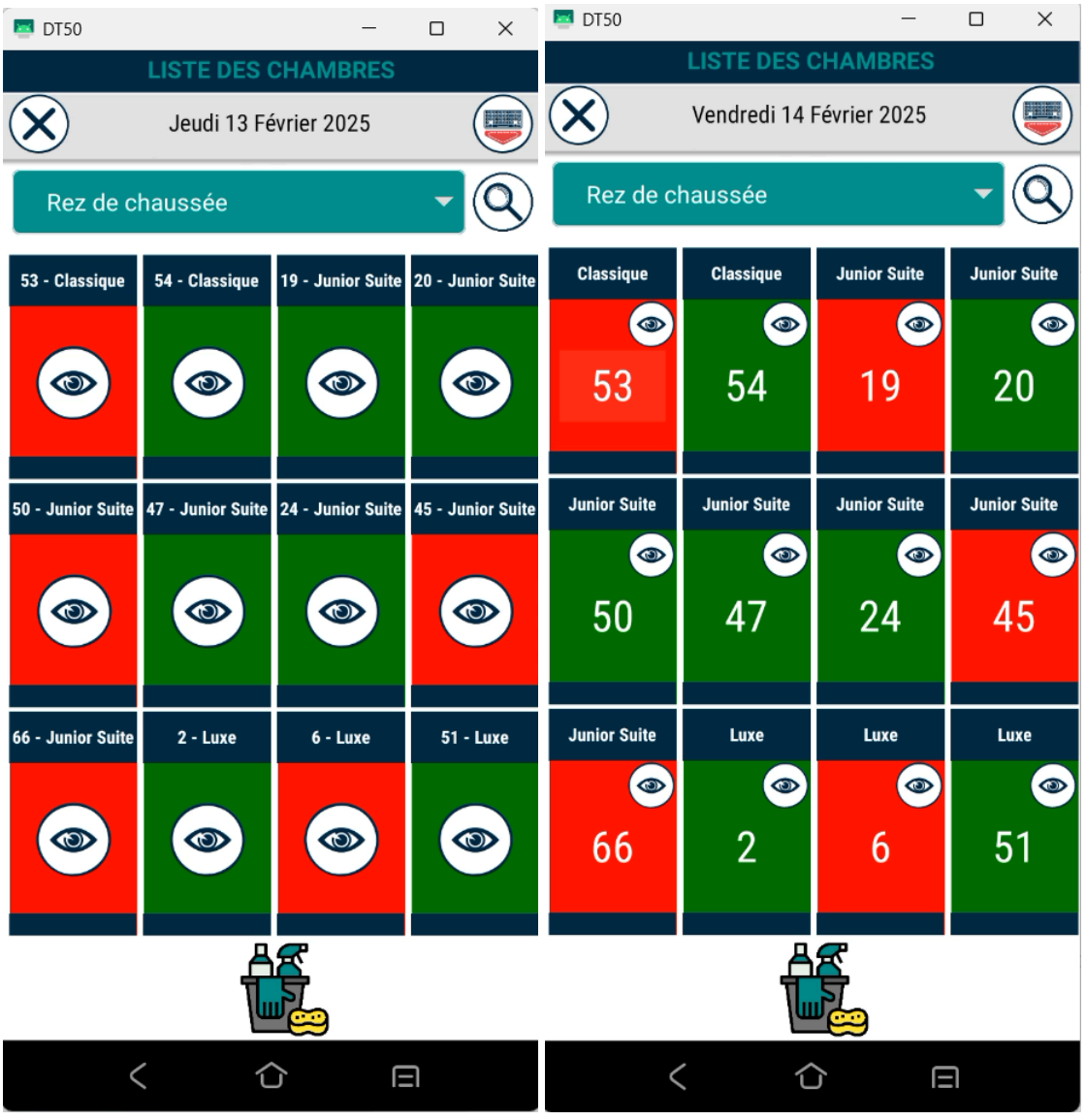
Nous arrivons donc sur le troisième plan de cette fenêtre sur lequel on peut retrouver le fameux champ pour écrire un commentaire en cas de problème, un bouton qui récupère la valeur de l'état libellé présente dans le Json, le fond du bouton est de la couleur de l'état actuel de la chambre et on a un

libellé qui l'accompagne permettant d’indiquer à l'utilisateur qu’il faut appuyer sur ce bouton afin de changer l’état de la chambre et aussi changer ce libellé pour afficher le fait que la chambre a bien été nettoyé si elle est propre.
Juste en dessous de cela, il y a normalement toute une section pour un système de photo mais que je n’affiche pas car il n’est pas encore fonctionnel.
Et il y a un bouton de validation en bas qui enverra les données saisies à la base locale et faire une update au niveau de l’API.



Comme vous pouvez le voir juste au dessus, quand on appuie sur le bouton du changement d'état, le bouton change de couleurs en devenant rouge, le libellé du bouton a quant à lui changé, on est passé de propre à sale et le libellé juste en dessous du message a aussi changé.

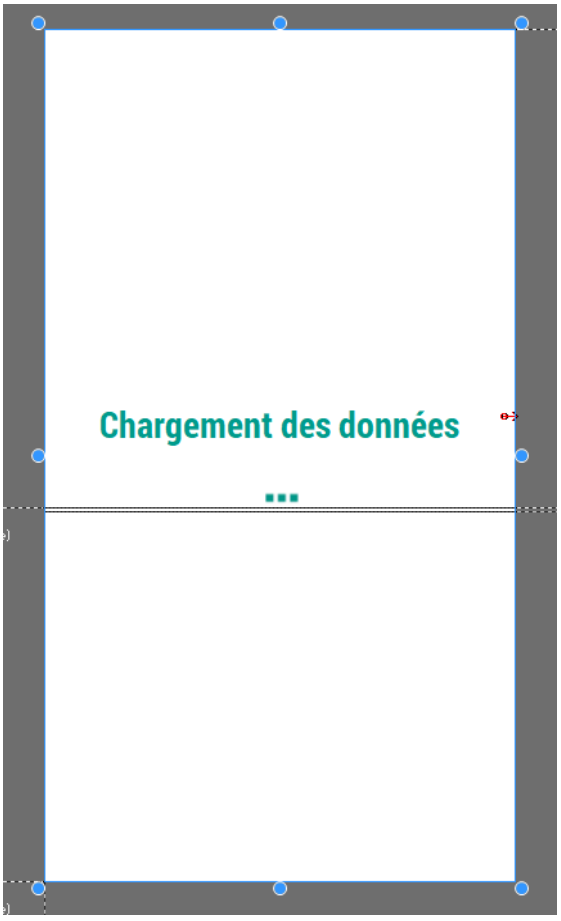
Si l'on revient sur le plan précédent depuis le bouton flèche en haut à gauche, les infos ne sont pas sauvegardé mais, si on clique sur le bouton de validation en bas à droite alors on peut donc remarquer que la couleur de fond de la chambre n°53 a bien été changé comme vous pouvez le voir ci-dessous :



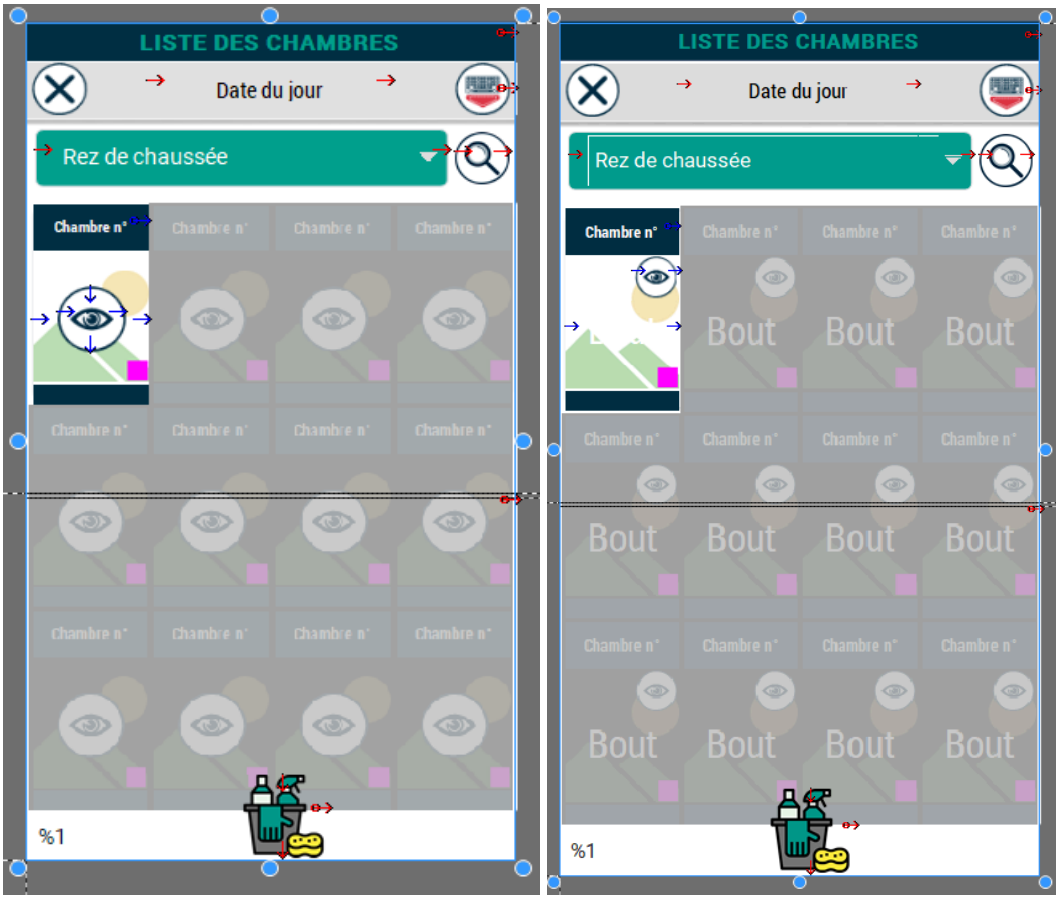
J’ai donc essayé de créer une interface qui soit la plus intuitive possible afin que tout le monde puisse s’en servir le plus facilement possible mais il se peut qu'à l'avenir, celle-ci soit de nouveau modifiée, cela dépendra des critiques que feront les utilisateurs.

Ce que j’ai fait sur le projet Windev Mobile (Présentation fenêtre WindevMobile)

Je vais maintenant vous montrer à quoi ressemble la fenêtre principale ff_entretiens sur ses différents plans avec les fenêtres internes que j’ai créé ainsi que toutes les variables que j’ai utilisé et on va commencer par le premier plan du chargement :



Il n’y a pas grand chose a dire sur ce plan, c’est juste un libellé nommé LiChargement et un Gif GifChargement qui s’affiche tant que toutes les les données sur les chambres ne sont pas chargées.



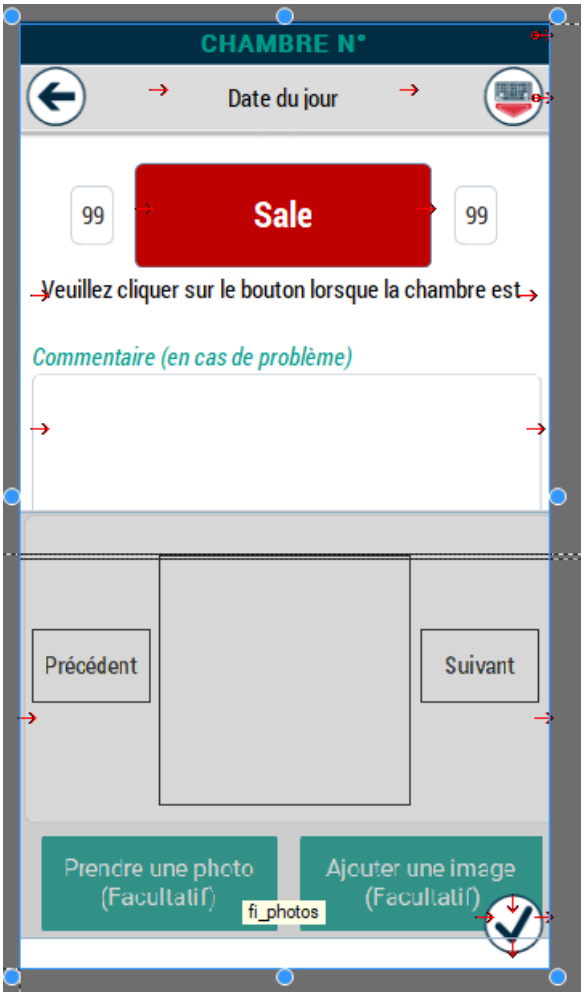
Au niveau du second plan, on aura beaucoup plus de détails que sur le plan précédent.
On peut donc voir le Combo CbEtageChambre qui permettra, une fois configuré, de choisir un étage en particulier et afficher les chambres qui en suivent, le bouton Rechercher BuRechercheChambre qui affiche le petite fenêtre Popup que l’on a vue précédemment, la fameuse zone répétée ZrListeChambres qui va permettre de dupliquer la première colonne non grisé en fonction du nombre de chambres que l’on

va récupérer dans le Json et ainsi d'afficher ainsi les bonnes informations à chaque chambres.
Pour chaque colonne des chambres dans la zone répétée, on aura plusieurs éléments propres à la chambre tels que une image ImgEtatChambre (AttEtatChambre) qui servira de fond pour chaque chambre et comme dit précédemment, le fond sera par défaut de la couleur de l'état actuel de chaque chambres, on a aussi le libellé LiNumeroChambre (AttNumeroChambre) qui quant à lui permet d'afficher le numéro de chambre, LiCategorieChambre(AttCategorieChambre) qui permet d'afficher la catégorie de la chambre et on a le bouton oeil BuChambre qui permet, quand il est appuyé de nous faire passer sur le troisième plan en affichant les détails correspondant à la chambre sélectionné.
On a le libellé LiDate qui permet d'afficher la date du jour, il y a le bouton BuRetour qui permet de retourner sur la fenêtre du menu principal une fois appuyé, le libellé LiTitre qui affiche un texte brut "Liste des chambres", une icône ImIconeEntretien et un libellé LiNbChambre qui est uniquement affiché en mode test afin de savoir directement le nombre de chambre affiché.

Il me reste juste les attributs que j'ai définie dans ma zone répétée mais qui sont uniquement appelés dans le code pour effectuer plusieurs actions (On n'est pas obligé de relier un Attribut a un élément de la zone répétée) qui sont AttEtatLibelle qui me permet de stocker le libellé de l'état de la chambre et donc effectuer plusieurs conditions en fonction, AttIdMarqueIndices qui quant à lui, me permet de récupérer l'identifiant web de la chambre afin de récupérer toutes ses informations, AttEtatChambre pour gérer la couleur de l'image de fond de chaque chambre de la zone répétée, AttNumeroChambre pour afficher tous les numéros de

chambres ainsi que leurs catégories, AttWebIdMarqueIndice pour stocker l'ID web de chaque chambres et AttTypeMarque qui permet de stocker l'ID du type de marque (son type, chambre, bar…) de chaque chambres .

Remarque : Dans une zone répétée, on peut définir ce que l'on appelle des Attributs (ce que j'ai mis entre parenthèses) qui vont nous permettre de pouvoir gérer l'affichage des différents éléments pour chaque chambre, on peut s'en servir si on souhaite effectuer une action sur une ligne et pas une autre.



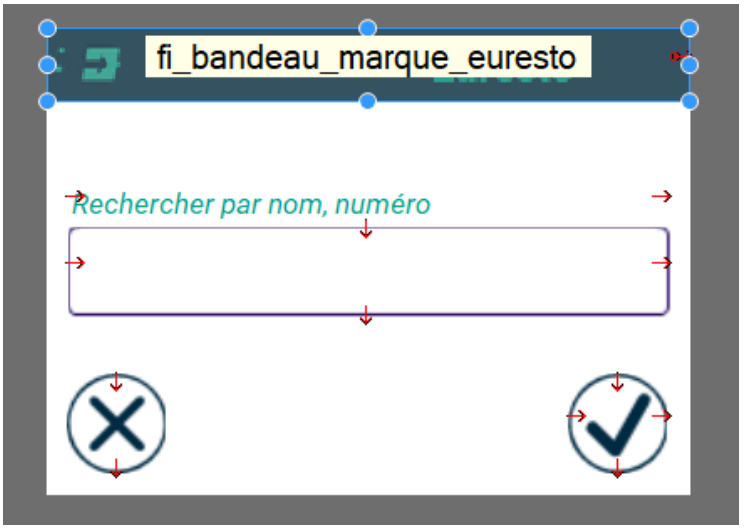
Au niveau du troisième et dernier plan, il y a aussi pas mal de choses à dire.
On a déjà le bouton en haut à gauche BuPrecedent qui permet juste de revenir au plan précédent (second plan), on revoit de nouveau la date affiché, le libellé situé tout en haut de la fenêtre étant LiChambre qui affiche le numéro de chambre ainsi que sa catégorie (exactement comme LiNumeroChambre dit précédemment), on a le champ de saisie SaCommentaire qui sert donc à écrire un commentaire, vous pouvez aussi remarquer la présence du bouton BuEtatChambre qui permettra de changer l'état de la chambre en un clic et si l'on s'est trompé ou que la chambre n'est finalement pas propre, on pourra alors réappuyer dessus afin de remettre la chambre dans son état initial et ainsi de suite.

Juste en dessous, on retrouve donc le libellé LiMessage qui change de message selon l'état de la chambre, on enchaîne avec une fenêtre interne fi_photos (que j'ai mis en invisible sur le projet) dont on a fait appel sur ce plan et à l'intérieur, on peut retrouver tout le système de photo que je vais vous présenter plus loin et finalement, si on regarde bien en bas à droite, on peut voir un bouton de validation BuValider qui permettra d'enregistrer en base locale les diverses informations saisies sur ce plan mais aussi au niveau des données web.

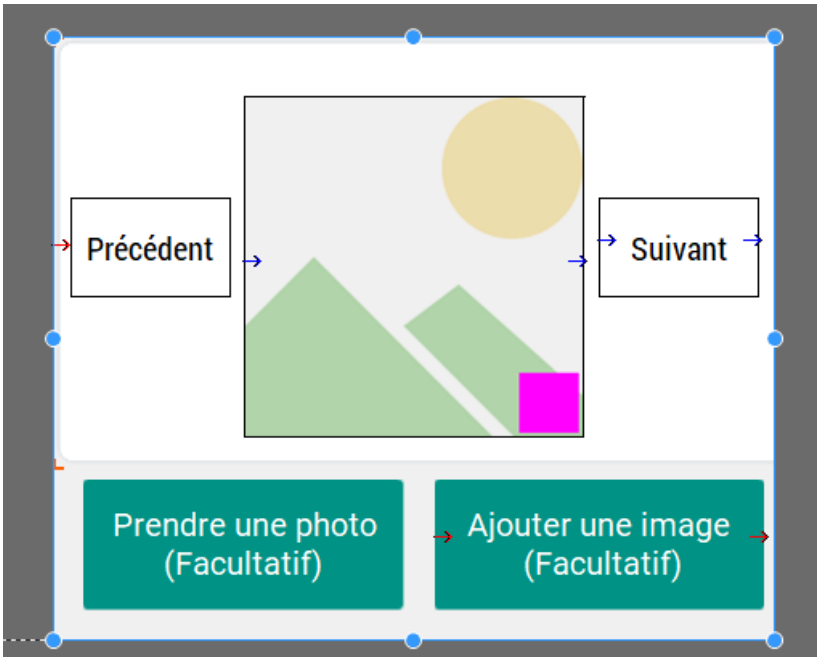
Il y a deux petits champs à côté du bouton étant SaEtatNumerique (à droite) et SaIdLocalTypeMarque (à gauche) qui sont uniquement affichés en mode test pour voir l'ID de l'état de la chambre et l'ID du type de marque de la chambre.

La prochaine fenêtre que nous allons voir est la fenêtre interne ft_entretiens_rechreche qui est la fenêtre qui s'affiche en forme

de Popup et qui nous permet d'effectuer la recherche des chambres que je vous affiche ci-dessous :



Ce n'est pas une très grande fenêtre comme l'on peut le constater, il y a quelques éléments tels que le bouton BuFerme qui permet juste de fermer cette fenêtre et revenir à la liste des chambres, le champ de saisie SaRechercheChambre qui permet d'écrire le numéro de chambre et le bouton BuValidation qui permet de vérifier le numéro de la chambre renseigné et ainsi d'afficher le troisième plan avec les détails de la chambre recherché.
Et il ne nous reste plus que la fenêtre interne fi_photos qui ressemble à ceci :



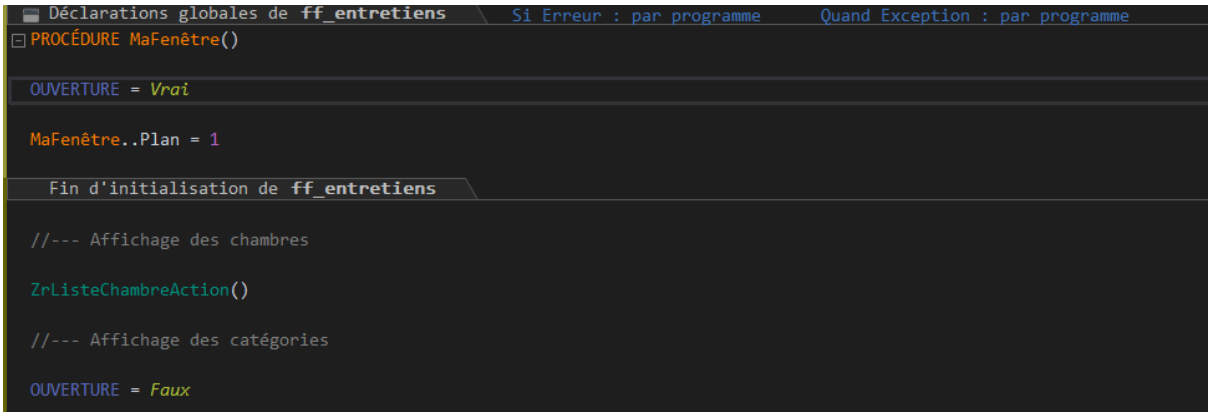
Vous pouvez donc y retrouver un champ Image ImPhotoChambre qui est transparent et dont on ne voit que le cadre extérieur qui permettra d'afficher la photo sélectionnée, on a le bouton en bas à gauche BuPrendrePhoto qui sert à ouvrir l'appareil photo du téléphone afin de prendre une photo et de l'afficher dans le champ Image, son libellé passe à "Prendre une autre photo", le bouton en bas à droite BuAjouterImage qui quant à lui permet de directement ouvrir les dossier des photos présents sur le téléphone afin de sélectionner une photo et aussi l'afficher sur le champ Image, son libellé passe à "Ajouter une autre image", il y a deux autre boutons à gauche et à droite du champ Image qui sont BuSuivant et BuPrecedent, le bouton de droite apparaît uniquement dès qu'une photo est affiché dans le champ Image et il permet de vider le champ Image pour pouvoir mettre une autre image tout en sauvegardant la précédente et le bouton de gauche quant à lui apparaît uniquement quand on appuie une fois sur le bouton "Suivant" et il permet de réafficher de nouveau l'image précédente et si on se retrouve sur la première image, alors il devient invisible.

Avant de passer à la partie suivante, je voudrais juste vous montrer à quoi ressemblent les informations Json que je récupère grâce à mon code, je ne mets que le début car la liste est vraiment très longue.
Elles ressemblent donc à ceci :


```
[
  {
    "id":19134,
    "libelle":"",
    "abrege":"     1",
    "actif":1,
    "key":"195737d6-a0f9-11ef-8145-00505639ca82",
    "type_marque_id":1,
    "capacite":1,
    "etage":null,
    "marque_categorie_id":657,
    "etat":"propre",
    "type_predefini_id":1,
    "correspondance_produits":null,
    "autorisation_modification":1,
    "entite_id":498373,
    "date_modification":"2025-01-23 14:19:46",
    "date_creation":"2024-11-12 14:22:00"
  },
  {
    "id":19135,
    "libelle":"",
    "abrege":"     2",
    "actif":1,
    "key":"19830b52-a0f9-11ef-8145-00505639ca82",
    "type_marque_id":1,
    "capacite":1,
    "etage":null,
    "marque_categorie_id":656,
    "etat":"propre",
    "type_predefini_id":1,
    "correspondance_produits":null,
    "autorisation_modification":1,
    "entite_id":498373,
    "date_modification":"2025-01-23 14:19:46",
    "date_creation":"2024-11-12 14:22:00"
  },
```

Ce que j’ai fait sur le projet Windev Mobile (Niveau Programmation)

Au niveau du code, je vais maintenant vous montrer toutes les choses que j’ai fait et on va commencer par le code de la fenêtre principale:



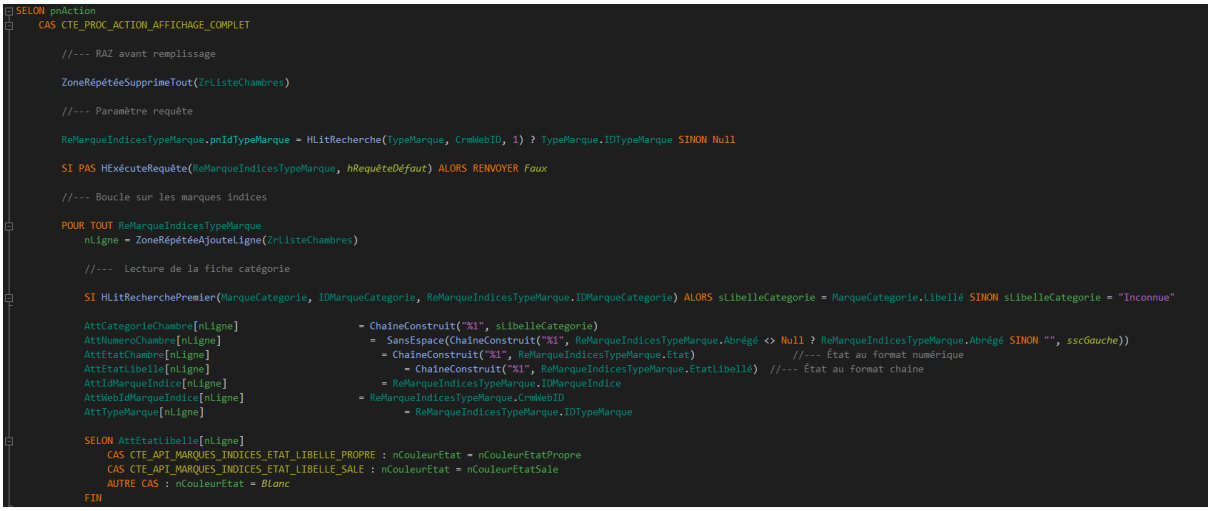
On commence donc avec l’initialisation de la fenêtre en définissant la procédure MaFenêtre() dans laquelle on initialise l’ouverture de cette fenêtre et on lui dit de commencer à afficher le premier plan, a la fin de l’initialisation, on fait appel à la procédure ZrListeChambreAction() que j’ai défini plus bas et on ferme l’ouverture.

```
PROCÉDURE ZrListeChambreAction(pnAction                    est un entier = CTE_PROC_ACTION_AFFICHAGE_COMPLET)

//--- Vars

nLigne                          est un entier
nCouleurEtat                    est un entier
nCouleurEtatSale                est un entier
nCouleurEtatPropre              est un entier

sLibelleCategorie           est une chaîne
sChambreEtatLibellé         est une chaîne
sMessageEtat                est une chaîne
sIconeChambre               est une chaîne


nCouleurEtatSale    = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_SALE, RougeClair)
nCouleurEtatPropre  = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_PROPRE, VertFoncé)
sIconeChambre       = ChargeParamètre(CTE_PARAM_ICONE_CHAMBRE,"")
```

On se retrouve maintenant dans la procédure ZrlisteChambreAction() qui est définie avec un paramètres qui est pnAction, un entier correspondant a la constante CTE_PROC_ACTION_AFFICHAGE_COMPLET, c'est ce qui va permettre d'afficher la liste de toutes les chambres.

Au niveau des variables, on en a plusieurs qui sont nligne, un entier qui permet de stocker le numéro de la ligne actuelle dans la liste des chambres, nCouleurEtat, nCouleurEtatSale et nCouleurEtatPropre qui sont des entiers permettant de stocker les différentes couleurs, sLibelleCategorie qui est une chaîne permettant de stocker le libellé de la catégorie de la chambre, sChambreEtatLibelé qui est une chaîne permettant d'afficher le libellé du bon état, sMessageEtat qui est une chaîne affichant le message sous le bouton selon l'état et SIconeChambre étant une chaîne qui sert à changer l'icône de l'oeil (voir plus bas dans Bonus).

On donne aux variables de couleurs et celle de l'icône la valeur qui est contenue sur ma fenêtre de paramètres.



```
SELON pnAction
    CAS CTE_PROC_ACTION_AFFICHAGE_COMPLET

        //--- RAZ avant remplissage

        ZoneRépétéeSupprimeTout(ZrListeChambres)

        //--- Paramètre requête

        ReMarqueIndicesTypeMarque.pnIdTypeMarque = HLitRecherche(TypeMarque, CrmWebID, 1) ? TypeMarque.IDTypeMarque SINON Null

        SI PAS HExécuteRequête(ReMarqueIndicesTypeMarque, hRequêteDéfaut) ALORS RENVOYER Faux

        //--- Boucle sur les marques indices

        POUR TOUT ReMarqueIndicesTypeMarque
            nLigne = ZoneRépétéeAjouteLigne(ZrListeChambres)

            //---  Lecture de la fiche catégorie

            SI HLitRecherchePremier(MarqueCategorie, IDMarqueCategorie, ReMarqueIndicesTypeMarque.IDMarqueCategorie) ALORS sLibelleCategorie = MarqueCategorie.Libellé SINON sLibelleCategorie = "Inconnue"

            AttCategorieChambre[nLigne]                 = ChaîneConstruit("%1", sLibelleCategorie)
            AttNumeroChambre[nLigne]                      =  SansEspace(ChaîneConstruit("%1", ReMarqueIndicesTypeMarque.Abrégé <> Null ? ReMarqueIndicesTypeMarque.Abrégé SINON "", sscGauche))
            AttEtatChambre[nLigne]                        = ChaîneConstruit("%1", ReMarqueIndicesTypeMarque.Etat)                  //--- État au format numérique
            AttEtatLibellé[nLigne]                          = ChaîneConstruit("%1", ReMarqueIndicesTypeMarque.EtatLibellé)  //--- État au format chaine
            AttIdMarqueIndice[nLigne]                     = ReMarqueIndicesTypeMarque.IDMarqueIndice
            AttWebIdMarqueIndice[nLigne]                = ReMarqueIndicesTypeMarque.CrmWebID
            AttTypeMarque[nLigne]                             = ReMarqueIndicesTypeMarque.IDTypeMarque

            SELON AttEtatLibelle[nLigne]
                CAS CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE : nCouleurEtat = nCouleurEtatPropre
                CAS CTE_API_MARQUES_INDICES_ETAT_LIBELLE_SALE : nCouleurEtat = nCouleurEtatSale
                AUTRE CAS : nCouleurEtat = Blanc
            FIN
```

Ensuite, on va dire que, selon pnAction, alors on va effectuer plusieurs constantes en commençant par la première qui est CTE_PROC_ACTION_AFFICHAGE_COMPLET, que l'on a défini plus haut.

On va d'abord vider la zone répétée des chambres avec ZoneRépétéeSupprimeTout() pour qu'a chaque exécution du code, les anciennes chambres ne s'incrémentent pas avec les nouvelles, on va ensuite rechercher un type de marque spécifique étant le 1 et exécuter une requête afin d'obtenir les chambres correspondantes.
Si l'exécution échoue, la procédure retourne Faux.
on va ensuite venir faire une boucle qui parcourt toutes les chambres disponibles dans ReMarqueIndiceTypeMarque (code SQL contenant les infos de MarqueIndice), on va alors ajouter une nouvelle ligne dans la zone répétée pour chaque chambre avec nligne.

On effectue ensuite une boucle qui va chercher la catégorie associée à la chambre actuelle et si la catégorie est trouvée,

elle est alors stockée dans sLibelleCategorie SINON, on lui donne une valeur de “Inconnu”.

On va ensuite mettre à jour les différents attributs avec les informations obtenues et une fois tout récupéré, on va afficher la bonne couleur de fond selon l’état de la chambre et la valeur de AttEtatLibelle.

```
        ZrListeChambres[nLigne].ImgEtatChambre.CouleurFond = nCouleurEtat
        ZrListeChambres[nLigne].Buchambre = Décode(sIconeChambre, encodeBASE64)
    FIN

    //--- fin requête

    HAnnuleDéclaration(ReMarqueIndicesTypeMarque)

    //--- Debug mode test uniquement

    SI gpbModeDeveloppement OU gpbModeTest ALORS
        (LiNbChambre..Visible, SaEtatNumérique..Visible, SaIdLocalTypeMarque..Visible)  = Vrai
        LiNbChambre = ChaîneConstruit(LiNbChambre, ZrListeChambres..Occurrence)
    FIN

    MaFenêtre..Plan = 2
```

On affecte par la suite les couleurs de fond définis dans les paramètres et aussi l’icône du bouton que l’on vient décoder.

On vient arrêter la déclaration de ReMarquendiceTypeMarque et, SI l’on est en mode développement ou test, ALORS on affiche les différents champs présents pour ces modes.

```
CAS CTE_PROC_ACTION_LIGNE_SELECTION //--- Sélection d'une chambre

    nLigne = ZoneRépétéeSelect(ZrListeChambres)
    SI PAS nLigne > 0 ALORS RENVOYER Faux

    //--- Valorisation des données sur plan 3

    SELON Minuscule(AttEtatLibelle[nLigne])
        CAS Minuscule(CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE)
            nCouleurEtat                = nCouleurEtatPropre
            sChambreEtatLibellé         = CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE
            sMessageEtat                = "La chambre a bien été nettoyée"
            SaIdLocalTypeMarque         = AttTypeMarque[nLigne]

        CAS Minuscule(CTE_API_MARQUES_INDICES_ETAT_LIBELLE_SALE)
            nCouleurEtat                = nCouleurEtatSale
            sChambreEtatLibellé         = CTE_API_MARQUES_INDICES_ETAT_LIBELLE_SALE
            sMessageEtat                = "Veuillez cliquer sur le bouton lorsque la chambre est propre"
            SaIdLocalTypeMarque         = AttTypeMarque[nLigne]

        AUTRE CAS
            nCouleurEtat                = Blanc
            sChambreEtatLibellé         = "État inconnu"
            sMessageEtat                = "Impossible de déterminer l'état de la chambre"
    FIN
```

On passe donc au cas suivant qui est CTE_PROC_ACTION_LIGNE_SELECTION qui va permettre d'effectuer une action selon une chambre en particulier.

On fait d'abord appel à nligne afin de pouvoir récupérer l'indice de la chambre sélectionnée puis, on effectue une condition qui va vérifier qu'une ligne est bien sélectionnée et si oui, alors on va venir récupérer les informations propres à la ligne sélectionnée et les affecter aux éléments du plan suivant.

On va ensuite venir vérifier avec une fonction que SELON AttEtatLibelle de chaque ligne, alors on va effectuer un des deux cas, le premier dans lequel la valeur du libellé est de CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE et pour lequel on va changer la couleur du bouton de l'état de la chambre et le libellé juste en dessous changera également et on récupère son Id local de type de marque.
Pour le second cas, dans lequel le libellé vaut CTE_API_MARQUES_INDICES_ETAT_LIBELLE_SALE, on va

alors affecter une couleur de fond au bouton, changer le libellé du dessous en fonction et on récupère son Id local de type de marque.

Pour les autres cas, on mettra des valeurs vierges.

```
//--- Affectation des valeurs / propriétés des champs

BuEtatChambre.CouleurFond       = nCouleurEtat
BuEtatChambre.Libellé           = sChambreEtatLibellé
LiMessage                       = sMessageEtat
LiChambre                       = AttNumeroChambre[nLigne]
SaEtatNumérique                 = AttEtatChambre[nLigne]

MaFenêtre..Plan = 3

SI sChambreEtatLibellé =  CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE ALORS
    SaEtatNumérique = CTE_API_MARQUES_INDICES_ETAT_PROPRE
SINON
    SaEtatNumérique = CTE_API_MARQUES_INDICES_ETAT_SALE
FIN
```

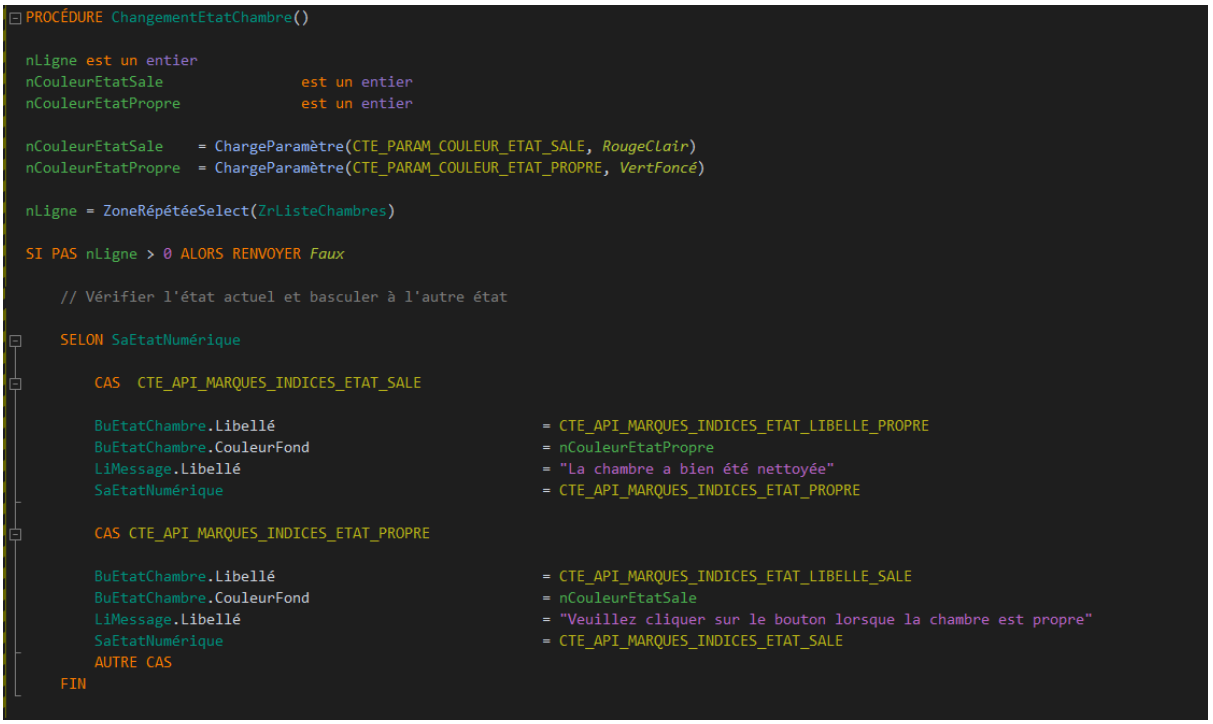
On affecte les valeurs aux différents champs, on passe au plan 3 et SI le libellé de l'état est propre, ALORS on donne l'ID de l'état propre à SaEtatNumérique et SINON, on lui donne la valeur de l'ID sale.

```
    CAS CTE_PROC_ACTION_MAJ_FIC //--- Validation des données / MAJ dans les tables

        nLigne = ZoneRépétéeSelect(ZrListeChambres)
        SI PAS nLigne > 0 ALORS RENVOYER Faux

        //--- Lecture de la fiche MarqueIndice

        SI PAS HLitRecherche(MarqueIndice, IDMarqueIndice, AttIdMarqueIndice[nLigne]) ALORS
            ToastAffiche("Un problème est survenu sur la lecture de la chambre")
            RENVOYER Faux
        FIN

        MarqueIndice.EtatLibellé = EtatChambreNumériqueVersLibellé(SaEtatNumérique)
        MarqueIndice.Etat        = EtatChambreLibelléVersNumérique(AttEtatLibelle[nLigne])

        //--- Si modification effectué en local, on met à jour la fiche MarqueIndice sur l'API

        SI HModifie(MarqueIndice) ALORS Mobile_ApiV2_Update_MarqueIndice(MarqueIndice.CrmWebID) SINON RENVOYER Faux

        //--- Actualisation des chambres

        ZrListeChambreAction()
        MaFenêtre.Plan = 2

    AUTRE CAS
FIN

//--- FIN OK

RENVOYER Vrai
```

On termine avec le dernier cas étant CTE_PROC_ACTION_MAJ_FIC qui permet de pouvoir faire en sorte que, lorsque les informations saisies sont validées, alors on viendra mettre à jour les différentes tables et données web.

On vient d'abord récupérer le numéro de la ligne sélectionné dans la zone répétée avec nligne, si la ligne n'est pas trouvée, on renvoie Faux, on va ensuite dire que Si la recherche de la chambre échoue, ALORS on affiche un message indiquant une erreur. On vient mettre à jour l'état de la chambre et Si des modifications ont été faites en locale, ALORS on vient mettre à jour sur l'API, on actualise la zone répétée et on retourne au plan n°2.

On a donc terminé avec tous les cas, on met donc fin à pnAction et on renvoie Vrai.

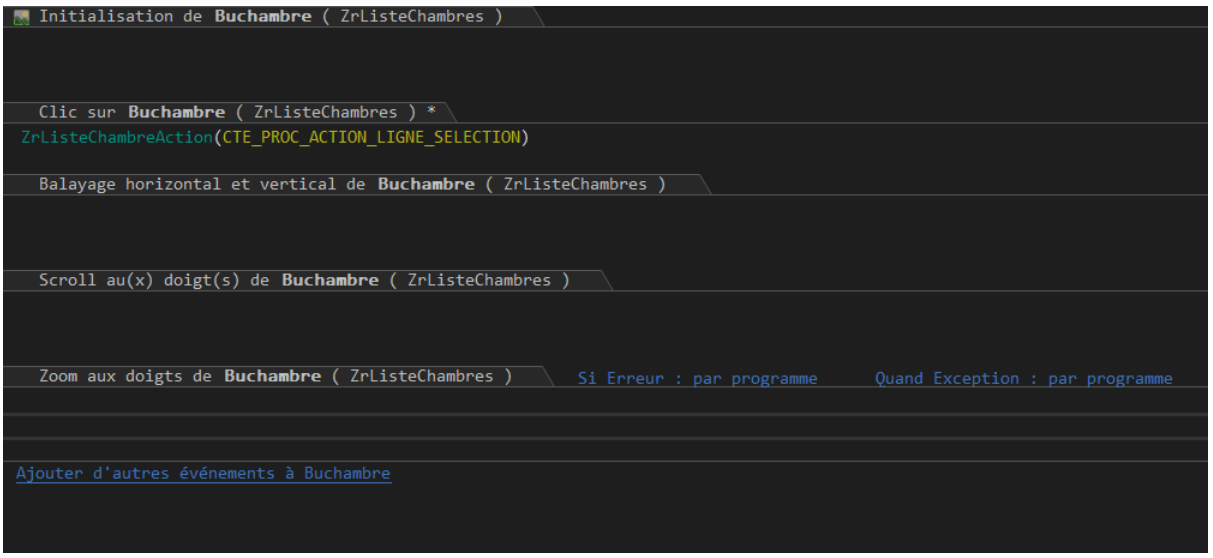


```
PROCÉDURE ChangementEtatChambre()

nLigne est un entier
nCouleurEtatSale              est un entier
nCouleurEtatPropre            est un entier

nCouleurEtatSale    = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_SALE, RougeClair)
nCouleurEtatPropre  = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_PROPRE, VertFoncé)

nLigne = ZoneRépétéeSelect(ZrListeChambres)

SI PAS nLigne > 0 ALORS RENVOYER Faux

    // Vérifier l'état actuel et basculer à l'autre état

    SELON SaEtatNumérique

        CAS  CTE_API_MARQUES_INDICES_ETAT_SALE

        BuEtatChambre.Libellé                       = CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE
        BuEtatChambre.CouleurFond                   = nCouleurEtatPropre
        LiMessage.Libellé                           = "La chambre a bien été nettoyée"
        SaEtatNumérique                             = CTE_API_MARQUES_INDICES_ETAT_PROPRE

        CAS CTE_API_MARQUES_INDICES_ETAT_PROPRE

        BuEtatChambre.Libellé                       = CTE_API_MARQUES_INDICES_ETAT_LIBELLE_SALE
        BuEtatChambre.CouleurFond                   = nCouleurEtatSale
        LiMessage.Libellé                           = "Veuillez cliquer sur le bouton lorsque la chambre est propre"
        SaEtatNumérique                             = CTE_API_MARQUES_INDICES_ETAT_SALE
        AUTRE CAS
    FIN
```

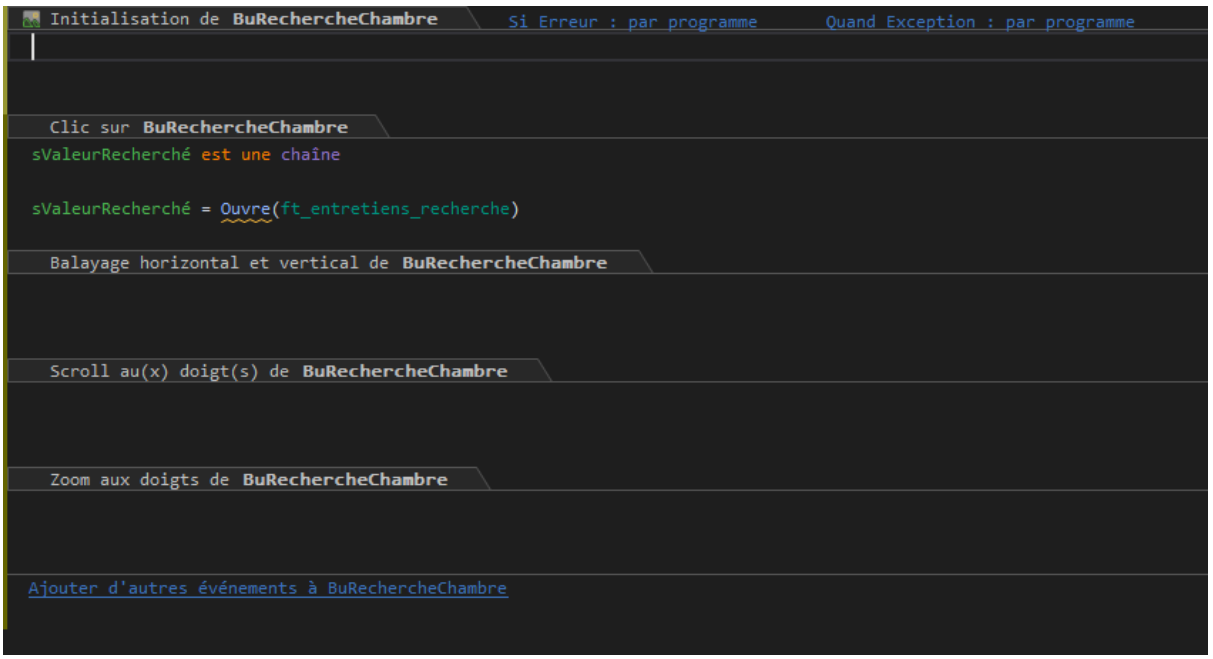
Nous voici maintenant dans la dernière procédure de ma fenêtre ff_entretiens qui est ChangementEtatChambre() et elle va me permettre de changer les couleurs et libellés des différents éléments en fonction de certains critères.

On vient tout d'abord définir nligne en tant que entier et on va ensuite lui donner la valeur de ZoneRépétéeSelect(ZrListeChambres) afin qu'il se positionne sur la ligne que l'on a sélectionné et on définit aussi nCouleurEtatSale et nCouleurEtatPropre étant aussi des entiers et ils ont là même fonction que montré plus haut.

On va ensuite venir vérifier que SELON la valeur de SaEtatNumerique (propre = 5 et sale = 4), on va alors effectuer deux CAS, le premier CAS représente l'ID de l'état sale et va permettre d'afficher toutes les infos correspondante à l'état propre et le second CAS, qui représente l'ID de l'état propre, va faire tout le contraire et si AUTRE CAS, alors on met FIN.



Au niveau du bouton en forme d'oeil, on va juste venir dans l'événement Clic du bouton dans lequel on va faire appel à notre précédente procédure étant ZrListeChambreAction() dans laquelle on va venir faire passer un paramètre qui est notre cas CTE_PROC_ACTION_LIGNE_SELECTION afin d'effectuer l'action sur la ligne de la zone répétée sélectionnée.



Au niveau du code du bouton BuRechercheChambre, on va aller dans l'événement Clic du bouton pour définir sValeurRecherché qui est une chaîne qui va juste nous permettre d'ouvrir la fenêtre interne ft_entretiens_recherche une fois que l'on appuyé sur le bouton.

```
Initialisation de BuValidation


   Clic sur BuValidation    Si Erreur : par programme    Quand Exception : par programme
  bTrouve                    est un booléen  = Faux
  nCouleurEtatSale           est un entier
  nCouleurEtatPropre         est un entier

  // Chargement des couleurs personnalisées
  nCouleurEtatSale    = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_SALE, RougeClair)
  nCouleurEtatPropre  = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_PROPRE, VertFoncé)

POUR i = 1 À ff_entretiens.ZrListeChambres..Occurrence
     ff_entretiens.ZrListeChambres = i

     sNumeroChambre est une chaîne = ExtraitChaîne(ff_entretiens.AttNumeroChambre, 1)

     SI SaRechercheChambre = SansEspace(sNumeroChambre) ALORS

         ff_entretiens.ZrListeChambres   = i

         ff_entretiens.Plan              = 3
         ff_entretiens.LiChambre         = ff_entretiens.AttNumeroChambre + " - " + ff_entretiens.AttCategorieChambre
         ff_entretiens.BuEtatChambre     = ff_entretiens.AttEtatLibelle
         bTrouve                         = Vrai
```

```
         // Application des couleurs dynamiques
         SI ff_entretiens.AttEtatLibelle = CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE ALORS
             ff_entretiens.ImgEtatChambre..CouleurFond   = nCouleurEtatPropre
             ff_entretiens.BuEtatChambre..CouleurFond    = nCouleurEtatPropre
             ff_entretiens.LiMessage..Libellé            = "La chambre a bien été nettoyée"
             ff_entretiens.SaEtatNumérique               = CTE_API_MARQUES_INDICES_ETAT_PROPRE
             ff_entretiens.SaIdLocalTypeMarque           = ff_entretiens.AttTypeMarque[i]
         SINON
             ff_entretiens.ImgEtatChambre..CouleurFond   = nCouleurEtatSale
             ff_entretiens.BuEtatChambre..CouleurFond    = nCouleurEtatSale
             ff_entretiens.LiMessage..Libellé            = "Veuillez cliquer sur le bouton lorsque la chambre est propre"
             ff_entretiens.SaEtatNumérique               = CTE_API_MARQUES_INDICES_ETAT_SALE
             ff_entretiens.SaIdLocalTypeMarque           = ff_entretiens.AttTypeMarque[i]
         FIN

         SORTIR
     FIN
FIN

SI bTrouve = Faux ALORS
     Info("Ce numéro de chambre est inexistant")
FIN

Ferme("", SaRechercheChambre)
```

Maintenant, nous sommes dans l'événement Clic du bouton de validation dans la fenêtre interne ft_entretiens_recherche dans

lequel on va commencer par définir bTrouve étant un booléen défini par défaut sur Faux qui servira à savoir si la chambre recherchée a été trouvée dans la liste ainsi que nCouleurEtatSale et nCouleurEtatPropre qui sont comme toujours des entiers pour gérer les couleurs que l'on vient d'ailleur récupérer juste après, on vient par la suite parcourir toutes les chambres affichées dans la liste ZrListeChambres grâce à la boucle Pour, ..Occurence nous permet d'obtenir le nombre total d'éléments dans la liste.

On sélectionne la ligne i de la liste pour accéder à ses données, on définit sNumeroChambre qui est une chaîne permettant d'extraire le numéro de chambre depuis AttNumeroChambre.

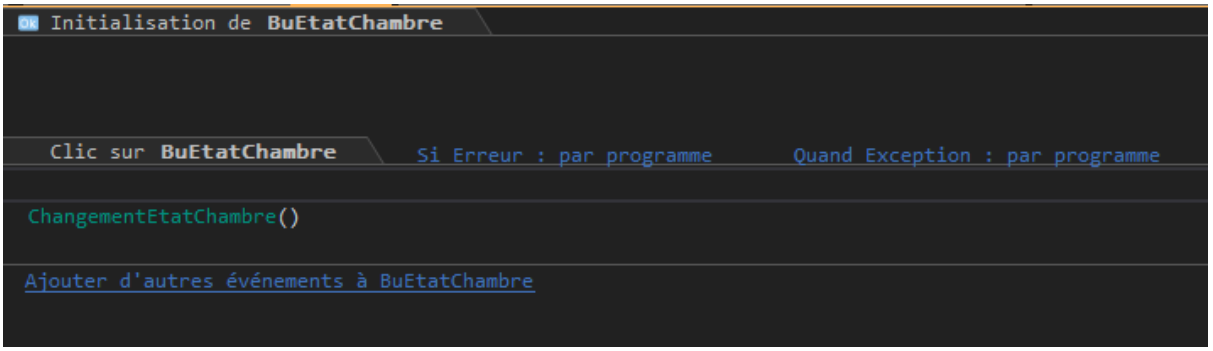
On va établir une condition disant que SI l'élément que l'on à écrit dans le champ de saisie SaRecherche correspond à un des numéros de chambres ou noms que l'on récupère avec l'attribut, ALORS on va de nouveau sélectionner la ligne i, on passe la fenêtre ff_entretiens au plan n°3, on met à jour les différents champs et on indique que la chambre à bien été trouvé en affectant à bTrouve la valeur Vrai.

Par la suite, on effectue une autre condition qui dit que, SI la valeur de AttEtatLibelle est de CTE_API_MARQUES_INDICES_ETAT_LIBELLE_PROPRE, ALORS on change tout les éléments pour que cela corresponde au libellé affiché et sinon, on change les éléments pour que cela corresponde au cas contraire. On peut donc sortir de la condition.

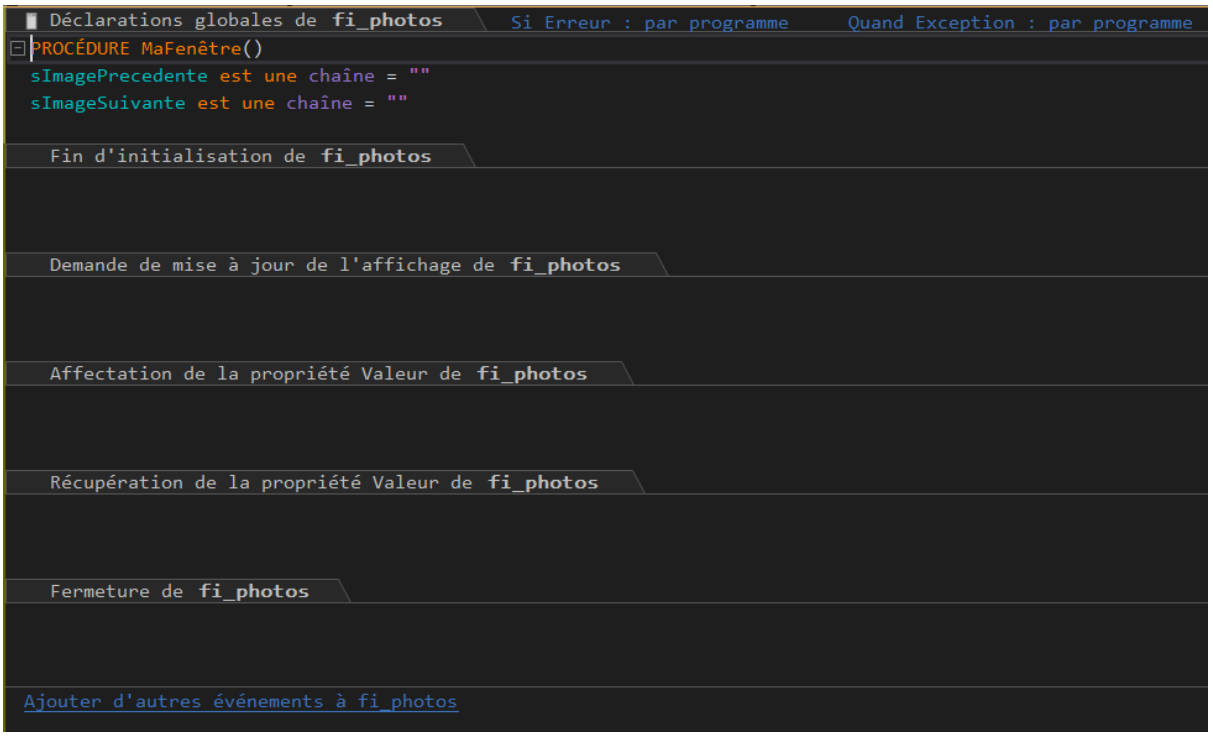
SI la valeur de bTrouve est égale à Faux, c'est à dire que l'on ne trouve pas le nom ou numéro de chambre renseigné,

ALORS on affiche un message d'information à l'utilisateur lui disant que l'information renseignée n'existe pas.

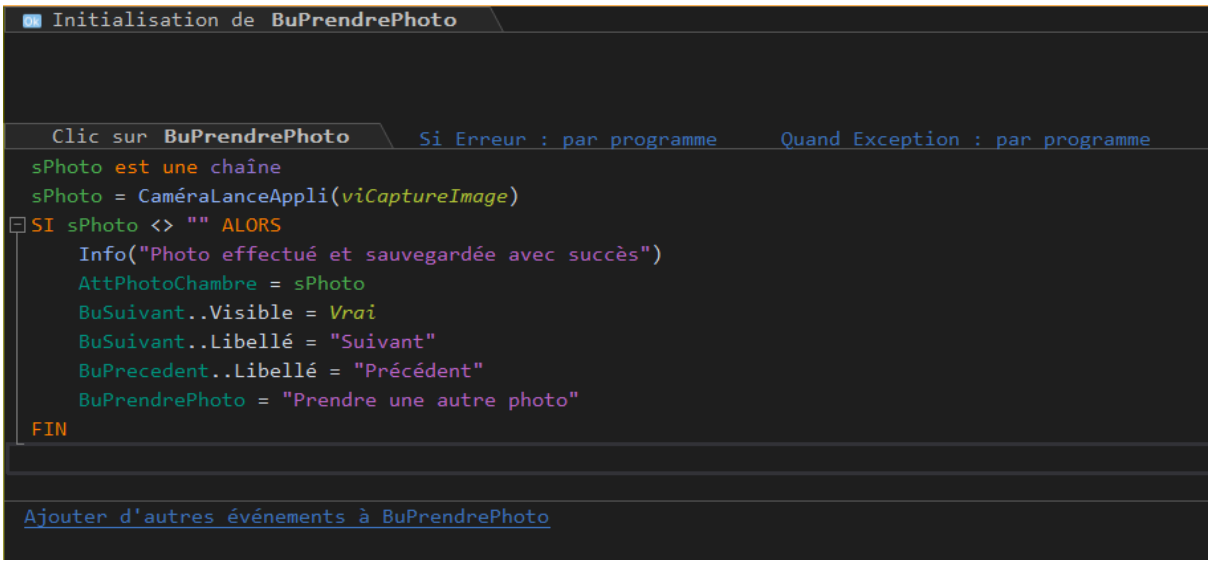
On finit par fermer la fenêtre et on indique SaRecherche en paramètre pour qu'il soit utilisé par la fenêtre qui l'a appelée.



Dans l'événement Clic du bouton BuEtatChambre, on vient juste appeler la procédure ChangementEtatChambre() afin de modifier les informations correspondantes.



On se retrouve dans le code de la fenêtre interne fi_photos (que j'ai mis en invisible) dans lequel on déclare juste deux variables qui sont sImagePrecedente et sImageSuivant qui sont toutes les deux des chaînes qui vont me permettre de gérer l'affichage de l'image dans le cadre.

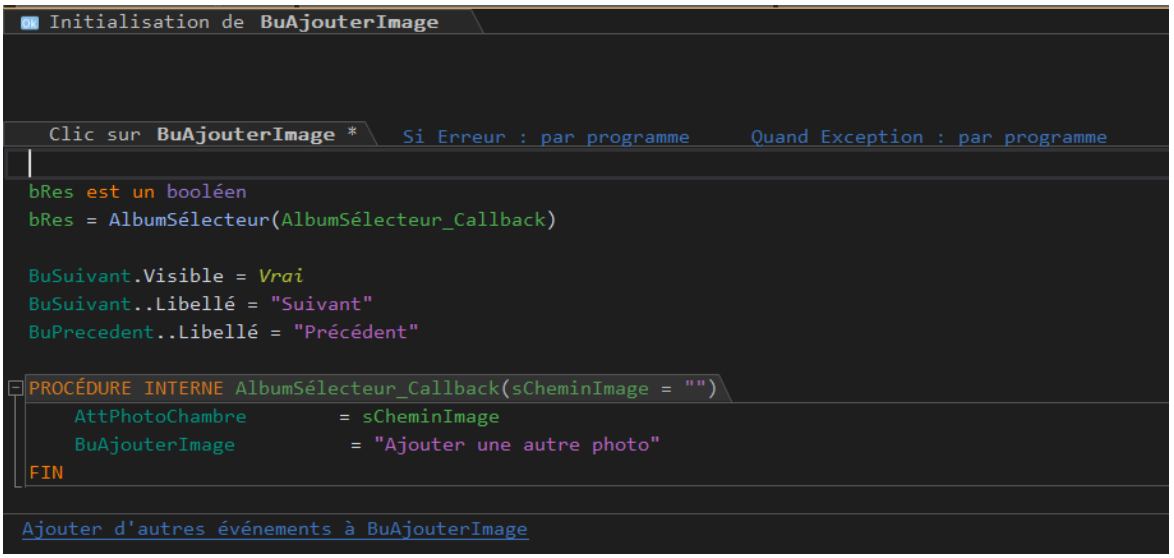


On est maintenant dans le code du bouton BuPrendrePhoto qui est le bouton me permettant d'ouvrir l'appareil photo du téléphone et de mettre par la suite la photo dans le cadre.

On va tout d'abord venir déclarer sPhoto qui sera une chaîne me permettant de stocker le chemin de l'image capturée, on vient ensuite dire que sPhoto va être égale à CaméraLanceAppli(viCaptureImage) et cela va me permettre d'ouvrir l'application Caméra afin de prendre une photo (viCaptureImage indique que l'on veut capturer une image et non une vidéo).

Et on va terminer par une condition qui dit que, SI sPhoto est différent de "" (rien), c'est à dire que l'on récupère une photo, ALORS on va faire plusieurs choses tels que informer

l’utilisateur du succès de la photo, on va afficher la photo dans l’attribut AttPhotoChambre qui représente le cadre ImPhotoChambre, on rend le bouton BuSuivant visible avec son libellé, on affecte aussi le libellé au bouton BuPrecedent et on finit par changer le libellé du bouton même afin d’indiquer à l'utilisateur qu’il peut prendre une autre photo.



On se retrouve dans le code du bouton BuAjouterImage qui me permet de sélectionner une photo dans l’album du téléphone afin de l’afficher dans le cadre.

On commence donc par déclarer bRes comme une variable de type booléen (Vrai ou Faux) qui va stocker le résultat de la sélection d'une image depuis l'album, on va ensuite lancer le sélecteur d'images du téléphone (galerie) avec la fonction AlbumSélecteur dans laquelle on va venir faire appel à la procédure AlbumSélecteur_Callback que j’ai définie plus bas, tout cela se fait si bRes retourne une valeur égale a Vrai.

On affiche donc le bouton BuSuivant avec son libellé et on affiche aussi le libellé du bouton BuPrecedent.

On vient ensuite définir la procédure interne AlbumSélecteur_Callback() qui sera appelée automatiquement quand une image est sélectionnée.
Elle reçoit en paramètre sCheminImage, qui contient le chemin du fichier image sélectionné.
A l'intérieur, on vient y stocker le chemin de l'image sélectionnée dans AttPhotoChambre et cela permettra de l'afficher ou l'envoyer plus tard et on finit par changer le libellé de ce même bouton afin d'indiquer à l'utilisateur qu'il peut ajouter d'autres photos si il le souhaite.

Ce que j'ai fait sur le projet Windev Mobile (Difficultées rencontrées)

Au niveau des difficultés, j'en ai rencontrés plusieurs que j'ai, pour la plupart, réussi à corriger qui sont :

Tout d'abord, j'ai eu des soucis au niveau de l'interface, du rendu visuel que j'allais faire car ce n'est pas forcément évident de savoir quel élément on va mettre à tel endroit, est ce que l'on met cette couleur ou une autre, est ce que cela rendrait pas mieux si on le positionne ici…

J'ai donc dû modifier plusieurs fois mon interface afin qu'elle corresponde finalement à ce dont on a exactement besoin.

Un autre problème que j'ai rencontré est le fait que lorsque je recherchais une chambre depuis la fenêtre interne ft_entretiens_recherche, il se pourrait qu'à certains moments, les éléments de détails de la chambres recherché soient de mauvaise couleur ou les infos affichés ne sont pas les bonnes.

Encore un problème que j’ai rencontré mais qui était juste un détail visuel était le fait que lorsque je voulais afficher l'abrégé de la chambre (son numéro), il y avait toujours un grand espace au début que je n’arrivais pas à retirer mais que j’ai finalement réussi à corriger en utilisant la propriété SansEspace() avec le paramètre sscGauche qui me permet donc de retirer tous les espaces étant présent au début de mon abrégé car, comme l’on peut le voir sur la capture d’écran du Json, il y a pour chaque abrégé un grand espace du côté gauche.

Un autre problème qui est d’ailleur le premier que j’ai rencontré est le fait que, au tout début du projet, j’avais un problème au niveau de la récupération des chambre car, dans le cas sur lequel je me suis occupé, je suis censé devoir récupérer 53 chambres mais moi, je récupérais plus de 280 chambres car je filtrais mal les données dans mon Json au niveau des identifiants des chambres appartenant à l’établissement concerné. Ce problème a bien été corrigé depuis.

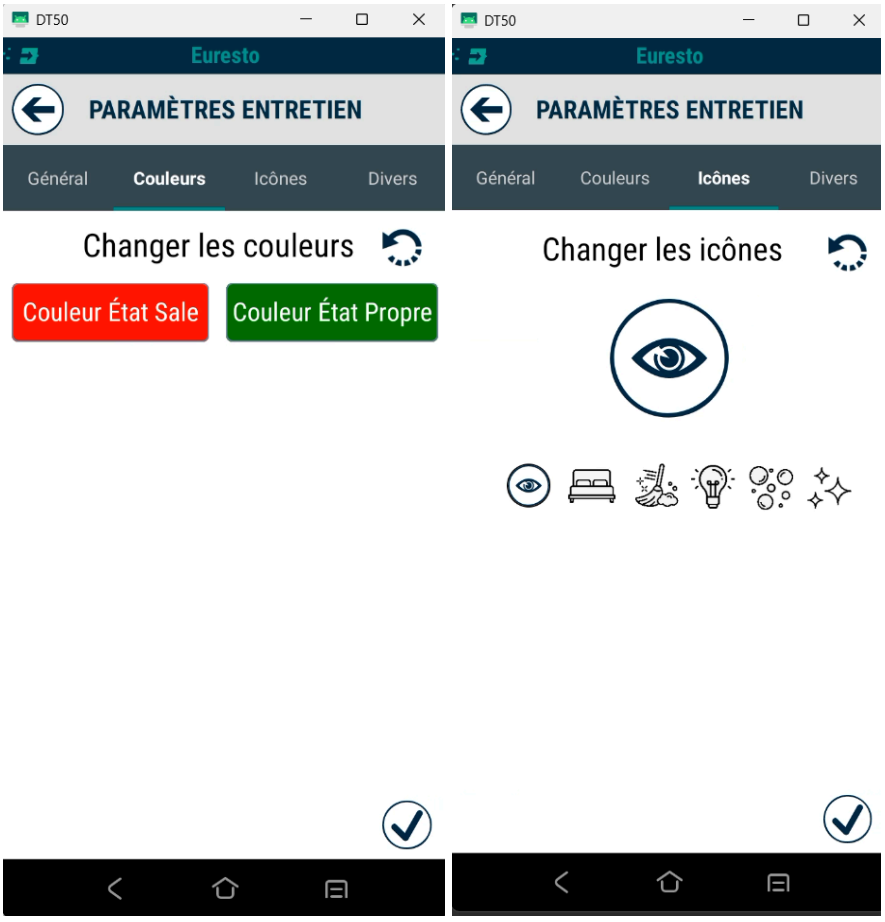
J’ai eu un autre problème au niveau de l’affichage du plan n°3 de la fenêtre selon la chambre que je choisis car ce qu’il se passait, c’est que quand je cliquais sur le bouton pour afficher les détails de la chambre sélectionné, j'affichais bien le bon plan mais les détails de la chambre sélectionné n’étais pas les bon, ils avaient tous comme numéro de chambre T9 qui correspondait à la toute dernière chambre présente dans le Json.
J’ai corrigé ce problème en disant dans le code de uniquement récupérer les détails de la chambre sélectionnés et pas toutes d’un coup.

Cela fait donc quand même pas mal de problèmes même si dans l'ensemble ils ont tous été corrigés.

Ce que j'ai fait sur le projet Windev Mobile (Bonus de fin de stage/Présentation)

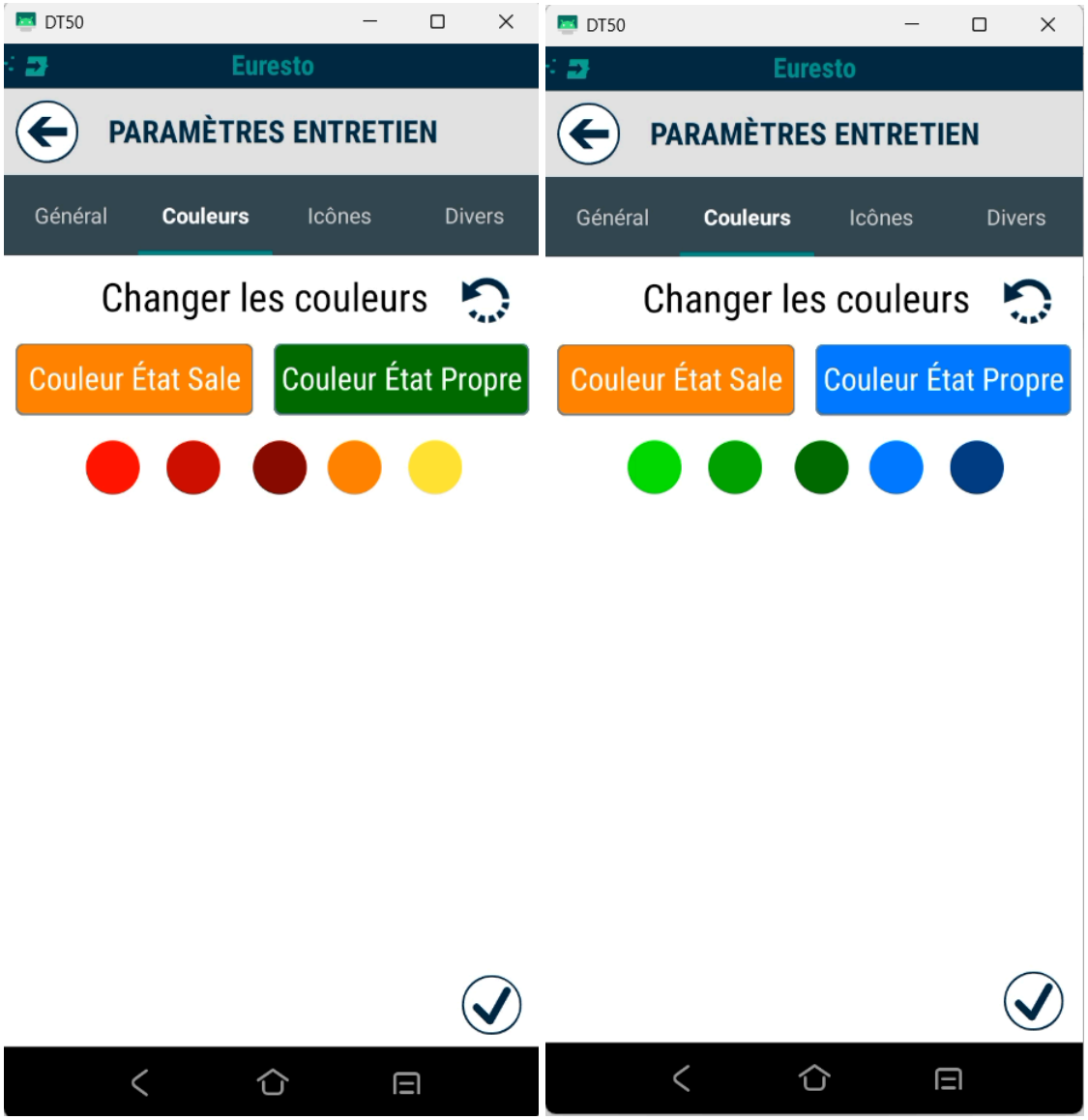
Pour la fin de mon stage, vu que j'avais fini le plus important, je me suis occupé de la création d'une nouvelle fenêtre nommée ff_getion_parametre_entretien qui regroupe des paramètres qui serviront uniquement pour la fenêtre que j'ai créé.
Cette fenêtre se trouve dans le second bouton que j'ai sur le menu principal.

Je vous la montre ci dessous :

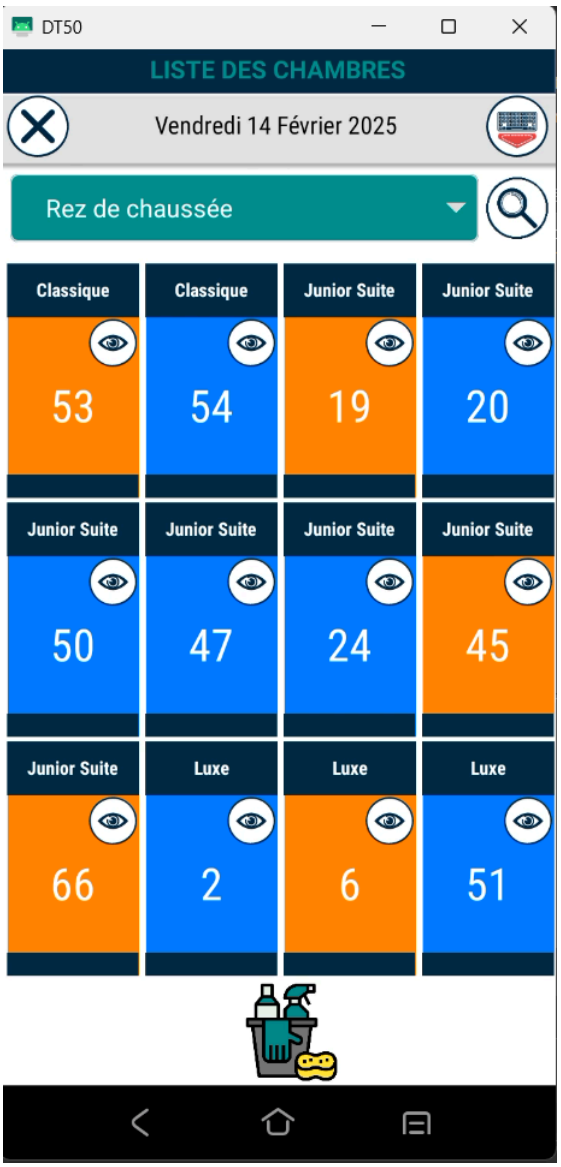


Voici ce dont à quoi ressemble la fenêtre, vous pouvez voir qu'il y a plusieurs onglets mais je n'ai malheureusement juste eu le temps de m'occuper de la section couleur qui permet de changer les couleurs des différents états et Icônes qui permet de changer l'icône dans la zone répétée. Vous aurez une flèche en haut à droite permettant de remettre les éléments à leur état initial.

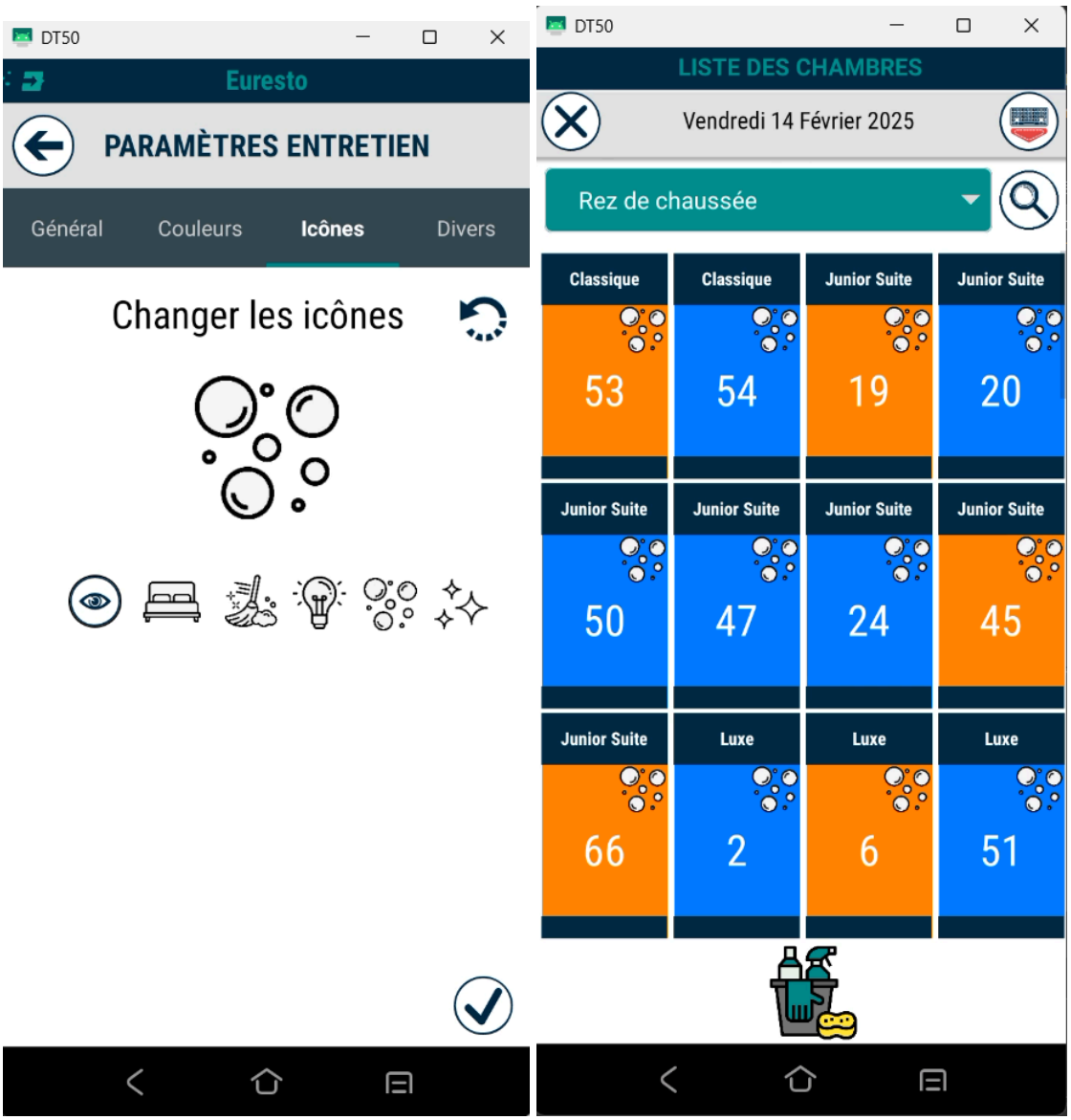
Selon l'état que vous allez sélectionnez, une palette de couleur qui ressort et vous pouvez changer la couleur comme ci dessous :



En fonction de la couleur que vous allez sélectionner, la couleur de l'état va donc changer et si l'on retourne sur la liste des chambres, on peut voir que les couleurs ont bien été changées.



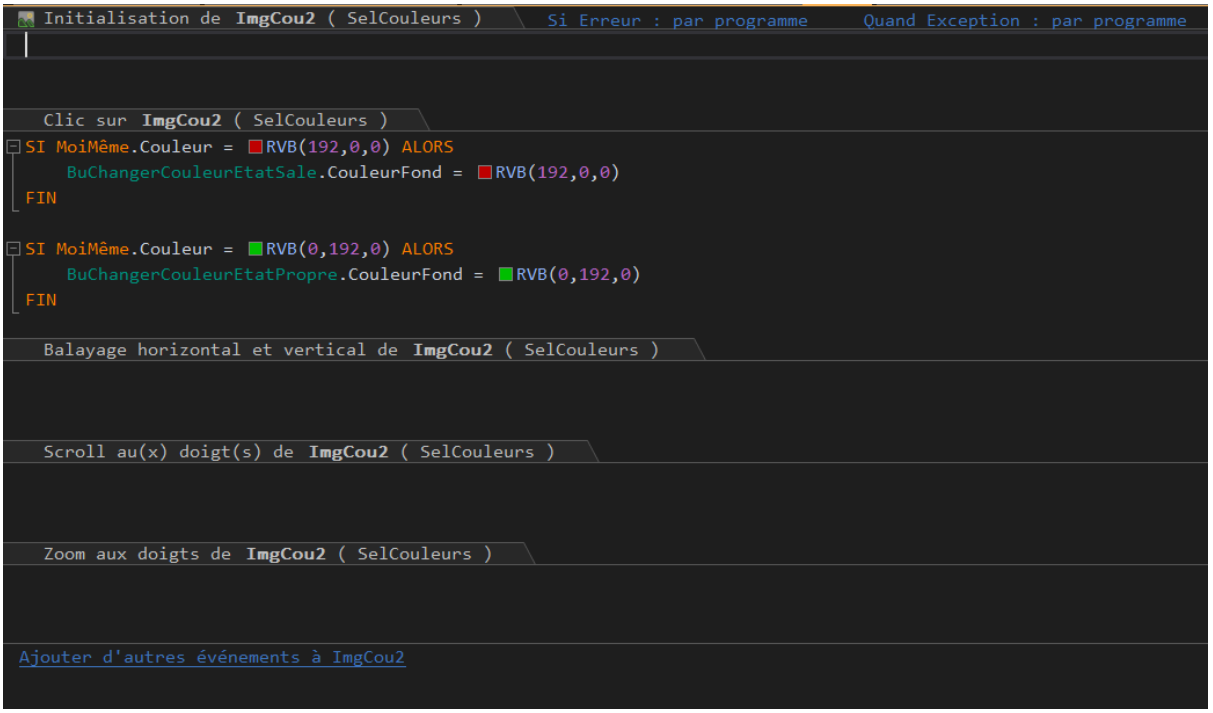
Au niveau de l'icône, cela va etre le meme fonctionnement comme vous pouvez le voir ci dessous :



Ce que j'ai fait sur le projet Windev Mobile (Bonus de fin de stage/Programmation)

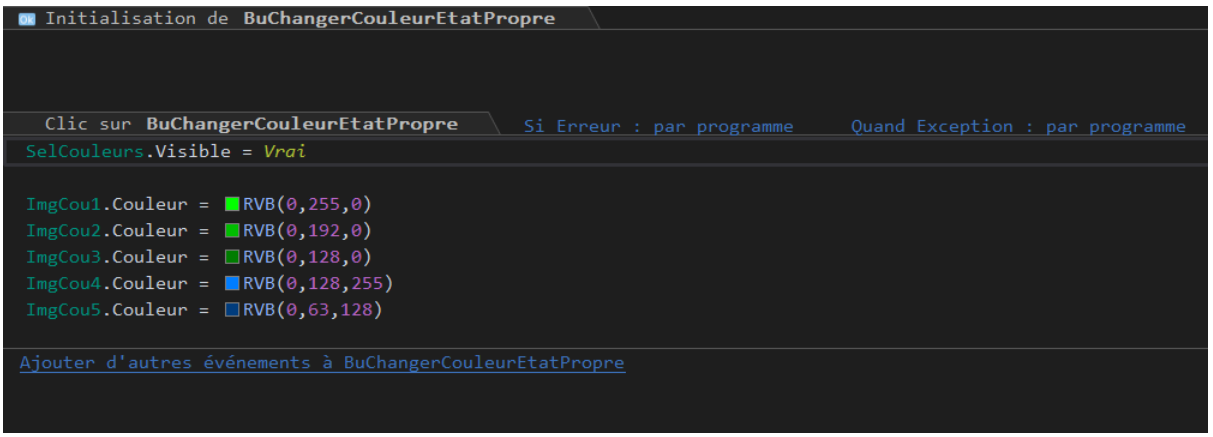
On va commencer avec le code de ImgCou2 (une des couleurs de la palette), ce code est identique pour chaque couleurs dans le sélecteur de la palette, il va juste vérifier que, SI la couleur de Moimême de ce code couleur rouge, ALORS le bouton BuChangerCouleurEtatSale prend cette couleur et on effectue

de nouveau la même condition pour le bouton BuChangerCouleurEtatPropre.

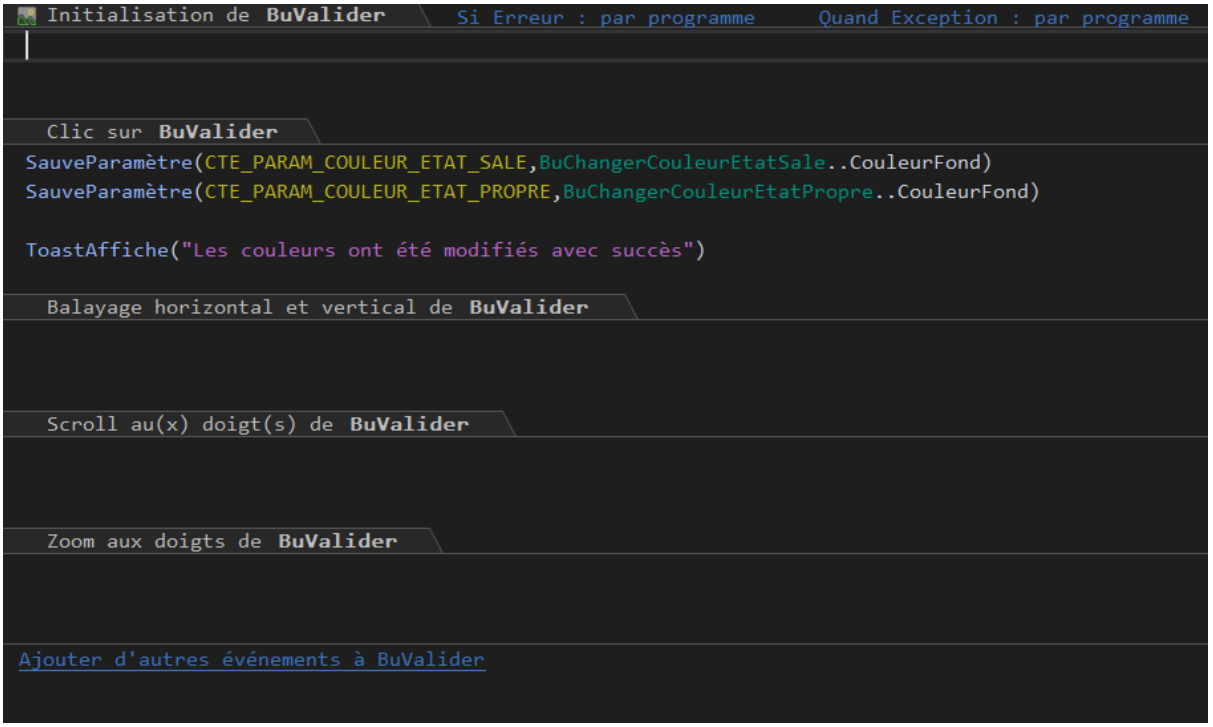


Au niveau du code du bouton BuChangerCouleurEtatSale, on dit juste au clic du bouton que l'on fait apparaitre le sélecteur de couleur et on met les couleurs correspondant à l'état sale.

Au niveau du code du bouton BuChangerCouleurEtatPropre, on fait strictement le même code que pour le bouton BuChangerCouleurEtatSale.



Dans le code du bouton BuValider, on va venir sauvegarder les différents paramètres que l'on a appliqués au niveau des couleurs grâce à SauveParamètre et on affiche un message indiquant le succès de l'action.



Dans le code de la fenêtre ff_gestion_parametre_entretien, je définis plusieurs variables tels que couleurSale et couleurPropre, des entiers qui stockent la valeur de la couleur et sImageSauvegardée qui est une chaîne permettant de charger l'icône utilisé, on vient récupérer la valeur actuelle de

chaque couleurs des différents états, on les remet aux couleurs initiales si non trouvés et on fait une condition qui dit que, SI sImageSauvegardée n'a pas de valeur, ALORS on vient affecter à ImgIconeChambre la valeur de sImageSauvegardée qui a été encodé précédemment en base 64.

```
Déclarations globales de ff_gestion_parametre_entretien    Si Erreur : par programme    Quand Exception : par programme
PROCÉDURE MaFenêtre()

couleurSale    est un entier
couleurPropre  est un entier
sImageSauvegardée est une chaîne = ChargeParamètre(CTE_PARAM_ICONE_CHAMBRE, "")

couleurSale                              = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_SALE, RVB(255, 0, 0)) // Valeur par défaut si non trouvée
BuChangerCouleurEtatSale..CouleurFond    = couleurSale

couleurPropre                            = ChargeParamètre(CTE_PARAM_COULEUR_ETAT_PROPRE, RVB(0, 128, 0)) // Valeur par défaut si non trouvée
BuChangerCouleurEtatPropre..CouleurFond  = couleurPropre

// Charge l'image sauvegardée


// Vérifie si une image était sauvegardée
SI sImageSauvegardée <> "" ALORS
    ImgIconeChambre = Décode(sImageSauvegardée,encodeBASE64)
FIN
```

Dans le bouton BuRetour, on vient juste faire un Ferme() pour fermer la fenêtre en cours d'exécution.

```
Initialisation de BuRetour    Si Erreur : par programme    Quand Exception : par programme


Clic sur BuRetour
Ferme()
```

Le code de BuCouleursInitiales permet juste, lors de l'événement Clic, de pouvoir réaffecter les couleurs initiales aux différents états.

```
Initialisation de BuCouleursInitiales


Clic sur BuCouleursInitiales    Si Erreur : par programme    Quand Exception : par programme
BuChangerCouleurEtatSale.CouleurFond      = RVB(255,0,0)
BuChangerCouleurEtatPropre.CouleurFond    = RVB(0,128,0)


Ajouter d'autres événements à BuCouleursInitiales
```

Le Code de BuImageInitiale permet de remettre l'image initiale s'il est cliqué.

```
Initialisation de BuImageInitiale

Clic sur BuImageInitiale    Si Erreur : par programme    Quand Exception : par programme
ImgIconeChambre = Img1
```

On termine avec le code d'une des icônes étant Img2 qui est identique pour toutes les autres dans lequel on va juste lui affecter sa propre valeur lors du clic a ImgIconeChambre.

```
Initialisation de Img2 ( SeImageBouton )    Si Erreur : par programme    Quand Exception : par programme

Clic sur Img2 ( SeImageBouton )
ImgIconeChambre = MoiMême
```

Et voici les 2 menus sous la vue en Windev.