Université Paris Cité

GESTIONNAIRE DE LOCATION DE VOITURE

KAYIBA CAMARA & BASTIDE DAVOUX

Introduction

Contexte

Dans un contexte économique où la gestion efficace des ressources devient essentielle pour assurer la rentabilité, les entreprises de location de véhicules sont confrontées à des défis quotidiens liés à la gestion de leurs opérations. Ces défis ne se limitent pas seulement à la maintenance des véhicules et au service client, mais s'étendent également à la gestion optimisée des réservations, des retours, et des paiements . Ce document détaille l'approche de notre application de gestion de location de véhicules, spécifiquement développée pour répondre aux besoins des gérants d'entreprises de location de voitures, localisées en France et de taille moyenne(ayant un répertoire de 100 clients pour un parc automobile de 50 à 100 véhicules). Cette solution sur mesure vise à transformer leur manière de gérer les opérations en intégrant des outils avancés et personnalisables.

Objectif

Ce document est conçu pour présenter une analyse détaillée de cette application de gestion de location de véhicules, une solution conçue sur mesure pour les gérants d'entreprises de location de voitures. L'objectif de cette application est de simplifier et d'optimiser les processus opérationnels en intégrant toutes les fonctions nécessaires au bon fonctionnement quotidien de telles entreprises. Tout en fidélisant la clientèle en leur proposant des avantages liés à cette fidélité totalement personnalisable par les propriétaires de ces structures.

Portée

Ce cahier des charges a pour objectif de décomposer de manière exhaustive tous les éléments de l’application du projet de location de véhicules. Cela comprend la prise en

charge complète de l’enregistrement des véhicules et des clients dans une base de données, la gestion des réservations et des locations, le suivi des retours de véhicules, la gestion des paiements, ainsi que la fourniture d’un service client entièrement personnalisable selon les besoins spécifiques de chaque entreprise.

L'application vise non seulement à améliorer l'efficacité des opérations quotidiennes mais également à renforcer la satisfaction des clients grâce à un système de fidélités ajustable par le propriétaire de la structure de location. Hormis la satisfaction de la clientèle, les gérants ne sont pas lésés par l’application. En effet, en plus de la gestion des services de bases liées à la location de véhicules, l’application propose un suivi des opérations de maintenance pour chaque véhicule et facilite la gestion des véhicules endommagés lors des locations. De plus, le système est conçu pour être évolutif, capable de s'adapter aux changements de taille et de stratégie de l'entreprise. L’application devra aussi fournir la possibilité de conserver les données des clients après leur suppression en cas de désistement et les données des véhicules supprimés. Bien entendu la conservation des données clients sera en accord avec les RGPD. Cela permettra de pouvoir réintégrer plus facilement les clients et leurs avantages après leur suppression en cas de désistement de leur résiliation,tout en nous permettant durant un temps limité, de continuer à démarcher et à inciter les anciens clients à réintégrer les agences utilisatrices de nos outils.

Outils

Ainsi ce cahier des charges explore également la modélisation des données nécessaires pour représenter fidèlement tous les aspects opérationnels de l'entreprise, l'architecture du système qui soutient ces processus, ainsi que l'examen des choix de conception et des technologies impliquées. En effet, toutes les informations client, véhicule et de location reposent sur une base de données en format texte (txt), facilitant la gestion, l'accès et la portabilité des données. Cette approche a été choisie pour sa simplicité et son efficacité, permettant une intégration aisée dans divers environnements et minimisant les exigences techniques pour les opérations de base de données.

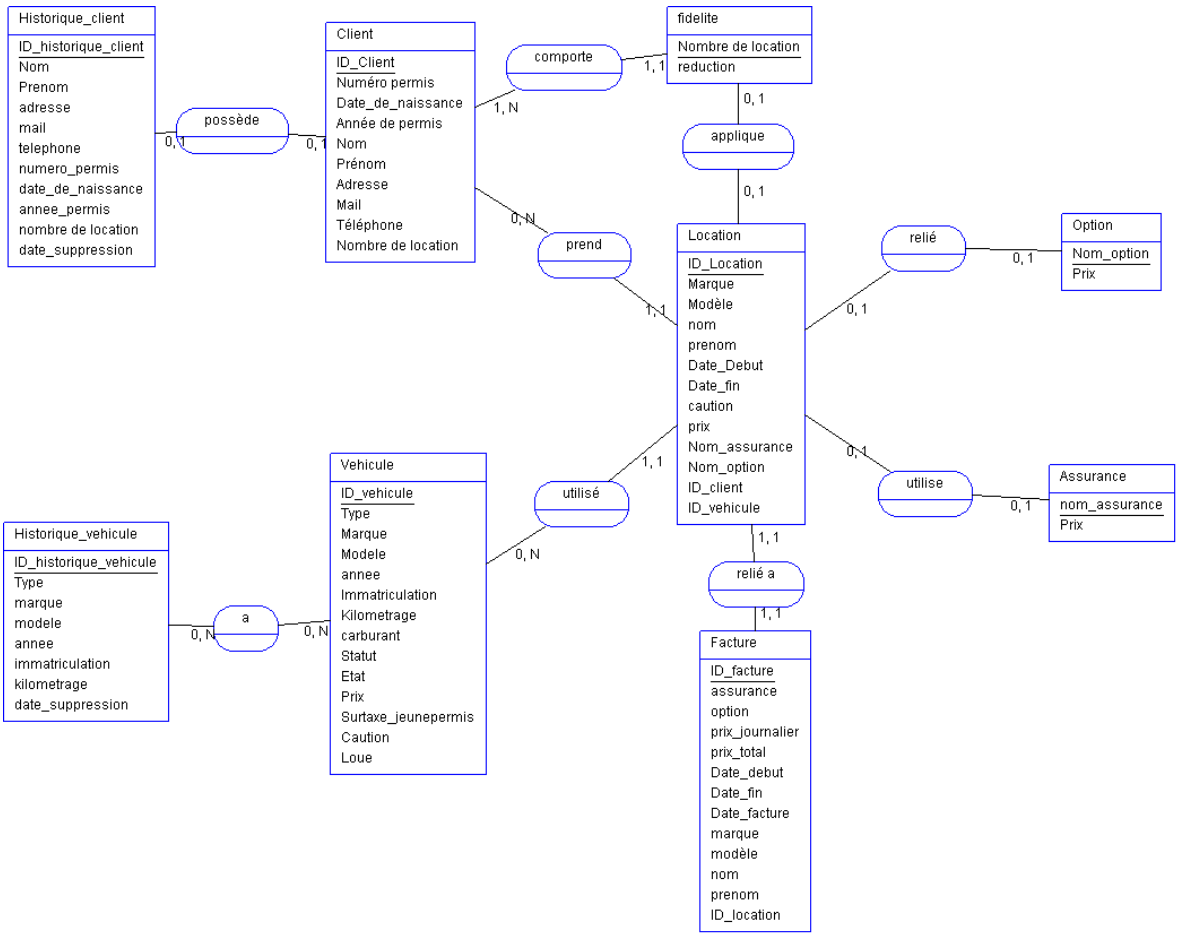
En somme, ce document est destiné à fournir une feuille de route claire pour le développement et la mise en œuvre de l'application, assurant que toutes les parties prenantes ont une compréhension claire et complète des objectifs, des fonctionnalités, et des bénéfices de l'outil proposé. L'utilisation du langage C pour le développement de l'application apporte plusieurs avantages significatifs, incluant la performance optimisée grâce à la gestion manuelle de la mémoire et une exécution plus rapide des opérations. De

plus, le langage C garantit la robustesse et la portabilité de l’application, permettant ainsi à l'application de fonctionner efficacement sur divers systèmes d'exploitation sans nécessiter de grandes modifications. Cette approche technique assure non seulement une base solide pour une application fiable et performante mais aussi une facilité de maintenance et d'évolution face aux besoins changeants du marché de la location de véhicules.

L'utilisation d’une base de données relationnelle nous oblige de nous assurer de la cohérence et de l’intégrité de cette dernière. Dans le but d’optimiser l’utilisation des données et dans un souci de cohérence et d'intégrité des données , la création d’un MCD et d’un MLD est nécessaire.

Le schéma conceptuel de données, également appelé MCD pour modèle conceptuel de données, est une représentation claire des données de la base de donnée à concevoir. Cette représentation en outre figure les relations entre les données.

MCD



Le MCD nous permet de visualiser les relations entre les différentes données et nous sert de guide lors de la réalisation de notre projet.

Le MLD est un autre outil schématique nécessaire à la création d'une base de données forte et cohérente. Le MLD (Modèle Logique de Données) organise les données selon des structures logiques.

MLD

Vehicule (ID_vehicule_Vehicule, Type_Vehicule, Marque_Vehicule, Modele_Vehicule, annee_Vehicule, Immatriculation_Vehicule, Kilometrage_Vehicule, carburant_Vehicule, Statut_Vehicule, Etat_Vehicule, Prix_Vehicule, Surtaxe_jeunepermis_Vehicule, Caution_Vehicule, Loue_Vehicule)

Client (ID_Client_Client, Numéro_permis_Client, Date_de_naissance_Client, Année_de_permis_Client, Nom_Client, Prénom_Client, Adresse_Client, Mail_Client, Téléphone_Client, Nombre_de_location_Client, #historique_client_id_historique_client_historique_client)

Location (ID_Location_Location, Marque_Location, Modèle_Location, nom__Location, prenom_Location, Date_Debut_Location, Date_fin_Location, caution_Location, prix_Location, Nom_assurance_Location, Nom_option_Location, ID_client_Location, ID_vehicule_Location, #ID_Client_Client, #ID_vehicule_Vehicule, #fidelite_nombre_de_location_fidelite, #facture_id_facture_facture, #option_nom_option_option, #assurance_nom_assurance_location)

Assurance (nom_assurance_Location, Prix_Assurance, #location_id_location_location)

Option (Nom_option_Option, Prix_Option, #location_id_location_location)

fidelite (Nombre_de_location_fidelite, reduction_fidelite, #ID_Client_Client, #location_id_location_location)

Facture (ID_facture_Facture, assurance_Location, option_Location, prix_journalier_Facture, prix_total_Facture, Date_debut_Facture, Date_fin_Facture, Date_facture_Facture, marque_Facture, modèle_Facture, nom_Facture, prenom_Facture, ID_location_Facture, #location_id_location_location)

Historique_client (ID_historique_client_Historique_client, Nom_Assurance, Prenom_Historique_client, adresse_Historique_client, mail_Historique_client, telephone_Historique_client, numero_permis_Historique_client, date_de_naissance_Historique_client, annee_permis_Historique_client, nombre_de_location_Historique_client, date_suppression_Historique_client, #client_id_client_client)

```
Historique_vehicule (ID_historique_vehicule_Historique_vehicule,
Type_Historique_vehicule, marque_Historique_vehicule,
modele_Historique_vehicule, annee_Historique_vehicule,
immatriculation_Historique_vehicule,
kilometrage_Historique_vehicule,
date_suppression_Historique_vehicule)
```

Ces concepts sont normalement utilisés lors de la création d'une base de données à l'aide d'un SGBD, néanmoins ces outils nous permettent de garantir une base de données claire et cohérente aux utilisateurs de notre application. Ainsi, malgré que notre projet soit uniquement programmé en langage C, nous avons jugé utile l'utilisation de ses concepts , afin de pouvoir concevoir et créer une base de données cohérente et efficace.

Fonctionnalités du Programme

Les besoins métiers

Lors de nos réunions avec l'équipe métier, nous avons retenu 8 fonctionnalités souhaitées par les métiers. Ils souhaitent de manière collégiale, que le programme réponde aux besoins suivants:

1. Gestion des détails de la location : Ils veulent une fonctionnalité qui leur permette de saisir en détail les informations pour chaque client ou véhicule.

2. Vérification de la disponibilité des véhicules : Ils veulent s'assurer que l'application vérifie automatiquement la disponibilité des véhicules pour les dates sélectionnées et propose des véhicules disponibles.

3. Gestion des tarifs et des options : Ils souhaitent pouvoir définir différents tarifs de location en fonction du type de véhicule, des dates de location, des options supplémentaires.

4. Traitement des paiements et factures : Ils demandent une fonctionnalité intégrée de traitement des paiements (calcul du coût de la location), gestion de la caution, ainsi que la génération automatique de factures.

5. Historicisation: Ils veulent pouvoir enregistrer les informations des clients, des véhicules et des locations et suivre l' historique des locations.

6. Reporting et suivi des performances : Ils veulent des outils de reporting pour suivre les performances de l'activité de location, comme les revenus générés, etc.

7. Sécurité et confidentialité des données : Ils veulent s'assurer que l'application garantit la sécurité et la confidentialité des données des clients, notamment en respectant les réglementations sur la protection des données personnelles RGPD.

8. Facilité d'utilisation et support utilisateur : Ils veulent une interface utilisateur intuitive et conviviale.

Pour répondre à ces besoins nous avons développé les fonctionnalités suivante:

2.1 Enregistrement des Véhicules et Gestion des Véhicules

Tout d'abord, dans notre programme nous avons définit un véhicule en utilisant les champs suivants:

num_vehicule: Il s'agit d'un identifiant unique pour chaque véhicule. L'intérêt d'avoir un numéro unique est de pouvoir identifier de manière unique chaque véhicule dans le système de gestion des locations. Cette valeur est incrémentée automatiquement pour éviter toute erreur
Type: Le type de véhicule (par exemple, SUV, berline, citadine, moto.).
Marque et modèle: La marque et le modèle du véhicule. Ces informations permettent de spécifier le véhicule de manière détaillée.
année: L'année de fabrication du véhicule.
Immatriculation: Le numéro d'immatriculation du véhicule. Cette information est essentielle pour l'identification du véhicule et sa traçabilité.
kilometrage: Le kilométrage du véhicule, indiquant le nombre de kilomètres déjà parcourus.
carburant: Le type de carburant utilisé par le véhicule. Nous avons des véhicules diesel(GAZ),essence(SP), électrique(ele)
statut: Le statut du véhicule, indique si le véhicule est disponible pour être loué ou s' il est en réparation, révision…
Etat: L'état général du véhicule, ce champ indique l'etat général du véhicule exemple: Bon, Mauvais, moyen.
prix: Le prix de la location du véhicule. Il s'agit du prix par jour TTC.
surtaxe_jeunepermis: Une surtaxe éventuelle est appliquée aux jeunes conducteurs en raison de leur manque d'expérience au volant.
caution: Le montant de la caution à déposer lors de la location du véhicule.
loue: Un indicateur du statut de location du véhicule (par exemple, "Oui" s'il est actuellement loué ou si une location est prévue dans le futur, "Non" si aucune location n'est prévue pour le moment pour le véhicule).

L'enregistrement et la gestion des véhicules constituent une composante essentielle de notre application de gestion de location de véhicules. Cette section détaille les fonctionnalités offertes par le système pour ajouter, modifier, et supprimer des informations relatives aux véhicules, ainsi que pour gérer la tarification et la disponibilité des véhicules.

2.1.1 Ajout de Véhicules

Les utilisateurs peuvent enregistrer de nouveaux véhicules dans le système en saisissant des informations détaillées nécessaires à leur gestion. Ces informations incluent :

- Marque : La marque du véhicule.
- Modèle : Le modèle du véhicule.
- Type: Le type du véhicule (citadine, 4x4,berline…)
- Année : L'année de fabrication du véhicule.
- Numéro d'immatriculation : Le numéro unique d'immatriculation du véhicule.
- Kilométrage : Le kilométrage actuel du véhicule lors de l'enregistrement.
- Type de carburant : Le type de carburant utilisé par le véhicule, par exemple essence, diesel, électrique, etc.
- Caution: L'utilisateur saisie la valeur de la caution
- Prix: L'utilisateur saisit le prix journalier de la location du véhicule.
- Surtaxe pour les jeunes permis : Si le véhicule expose les jeunes permis a une surtaxe ainsi que la valeur de cette dernière.

Ces informations sont essentielles pour non seulement identifier le véhicule mais aussi pour définir ses caractéristiques spécifiques qui peuvent influencer la décision de location par les clients.

les valeurs suivantes sont automatiquement incrémentée à l'ajout d'un véhicule dans l'application

- **Loue** prend la valeur "non"
- **Etat** prend la valeur "Bon"
- **Statut** prend la valeur "disponible"

Ainsi nous partons du principe qu'un véhicule nouvellement saisi est prêt à la location et est en bon état. Si cela n'est pas le cas alors l'utilisateur peut modifier la saisie dans la rubrique "modification du véhicule".

2.1.2 Modification et Suppression des Véhicules

L'application permet la modification des détails d'un véhicule en cas de mise à jour des informations comme le kilométrage, le type de carburant après une modification, ou simplement une erreur initiale lors de l'enregistrement. Les utilisateurs peuvent également supprimer un véhicule du système lorsque celui-ci n'est plus disponible ou en état de location, garantissant ainsi que la base de données reste à jour et précise.Les véhicules supprimés sont stockés dans un historique.

2.1.3 Fixation du Prix de Location

Chaque véhicule peut avoir un prix de location différent basé sur des facteurs tels que sa marque, son modèle, son année, et d'autres caractéristiques spécifiques. Les gérants peuvent fixer et ajuster le prix de location directement via l'interface de l'application, permettant une gestion tarifaire flexible et réactive aux conditions du marché.

2.1.4 Saisie de la Caution

La caution est un élément crucial pour protéger les intérêts financiers de l'entreprise de location. L'application permet de saisir et de modifier le montant de la caution pour chaque véhicule, ce qui est indispensable pour couvrir les éventuels dommages ou pertes pendant la période de location.

2.1.5 Affichage des Véhicules

L'application offre une vue d'ensemble des véhicules disponibles pour la location, ainsi que ceux qui sont actuellement en location ou en réparation. Ce module permet de visualiser l'état actuel de chaque véhicule (disponible, en cours de location, en réparation), aidant ainsi les gestionnaires à prendre des décisions informées concernant la gestion de la flotte et la planification des réservations. Pour améliorer la visualisation des véhicules, l’application propose d'afficher les véhicules par types, pour augmenter l’efficacité de la recherche des données d’un véhicule en particulier.

De plus, l'application est capable de pouvoir afficher les véhicules récemment supprimés.

Ces fonctionnalités sont implémentées dans le code source en langage C et sont soutenues par des fichiers de données en format texte, assurant une gestion simple et efficace des données des véhicules. Le système conçu permet non seulement une manipulation aisée de ces informations mais aussi leur récupération rapide pour toutes formes de consultations ou de rapports nécessaires à la bonne conduite des activités de l'entreprise.

2.2 Enregistrement des clients et Gestion des clients

La structure client est conçue pour stocker des informations sur les clients qui louent des véhicules. Voici une explication détaillée des différents champs :

num_client: Un identifiant unique pour chaque client. L'intérêt d'avoir un numéro unique est de pouvoir identifier de manière unique chaque client dans le système de gestion des locations. Cela permet de faciliter la recherche et la gestion des informations clients. Cette valeur s'incrémente automatiquement, pour éviter toute erreur.
num_permis: Le numéro de permis de conduire du client. Dans le contexte de la location de véhicules, il est important de collecter cette information pour vérifier si le client est autorisé à conduire le véhicule loué.
date_de_naissance: La date de naissance du client. Cette information peut être utilisée pour vérifier l'âge du client, ce qui est souvent un critère important dans la location de véhicules. Par exemple, certains loueurs imposent des restrictions d'âge pour la location de certains types de véhicules.
annee_permis: L'année d'obtention du permis de conduire. Cette information peut être utilisée pour évaluer l'expérience de conduite du client. Dans certains cas, les loueurs peuvent appliquer des tarifs différents en fonction du nombre d'années depuis l'obtention du permis.
nom et prenom: Les nom et prénom du client.
Adresse, mail et téléphone: Les informations de contact du client. Elles sont utiles pour communiquer avec le client et pour l'envoi de documents relatifs à la location.
nblocation: Le nombre de locations effectuées par le client. Cette information est utile pour évaluer la fidélité du client et ainsi lui offrir des réductions.

2.2.1 Ajout client

Les utilisateurs peuvent enregistrer de nouveaux clients dans le système en saisissant des informations détaillées nécessaires à leur gestion. Ces informations incluent :

- Données personnelles : L'utilisateur doit entrer le nom, le prénom, l'adresse, le numéro de téléphone, le mail et la date de naissance du client. Ces données permettent d'obtenir les données de contact du client et de pouvoir l'identifier.

- Données permis: L'utilisateur doit entrer le numéro de permis du client et la date d'obtention du permis. Ces données permettent d'établir si le client est un jeune permis. Ces données sont essentielles lors d'une inscription dans une agence de location. De plus, si le client se rend coupable d'une infraction routière, l'amende sera directement redirigée vers son adresse et les points prélevés sur son permis.

Le nombre de locations est automatiquement initialisé à 0. Nous partons du principe qu'un nouveau client n'a pas pu faire de location en amont de son inscription.

2.2.2 Modification et Suppression des clients

L'application permet la modification des détails d'un client en cas de mise à jour des informations comme le numéro de téléphone, le nom , ou simplement une erreur initiale lors de l'enregistrement. Les utilisateurs peuvent également supprimer un client du système lorsque celui-ci souhaite quitter l'agence, garantissant ainsi que la base de données reste à jour et précise.

2.2.3 Historisation et RGPD

L'application devra permettre d'historiser les clients supprimés. La conservation de ces données devra permettre aux agences de pouvoir continuer à démarcher les clients après leurs suppressions ou dans le cas où le client voudrait réintégré l'agence après l'avoir quitté tout en conservant ses avantages liés à la fidélité. La conservation de ses données sera limitée à un mois pour être en accord avec la loi en vigueur. Ainsi l'application à la capacité de pouvoir réintégrer dans la base de données les clients supprimée depuis moins d'un mois.

2.2.4 Affichage des Clients

L'application offre une vue d'ensemble des données clients enregistrés dans l'agence. En plus de cette vue d'ensemble, l'application propose de sélectionner plus précisément les données d'un client en procédant par la recherche de son numéro client. L'affichage de l'historique est également possible permettant de pouvoir obtenir une vue d'ensemble de l'intégralité des données des clients supprimées.

2.3 Enregistrement et utilisation de la Fidélité

Notre programme de fidélité récompense les loueurs fréquent, plus vous louez, plus vous grimpez les échelons de fidélités vous donnant le droit à des réductions prédéfinies par l'agence de location. En plus de fidéliser la clientèle, ce système les poussent à la consommation.

2.3.1 Initialisation fidélité

L'application propose à l'utilisateur de personnaliser son propre programme de fidélité reposant sur le principe précédemment expliqué. Ainsi dans la méthode "saisie_fidelite", l'utilisateur a le choix d'instaurer 10 échelons de fidélité. L'utilisateur définit pour chaque niveau le nombre de location a faire pour atteindre l'échelon et le taux de réductions que l'échelon apporte.

2.3.2 Automatisme et fonctionnement de la fidélité

La réduction liée à la fidélité s'applique de façon automatique au prix lorsque le client est éligible. Le programme utilise la valeur contenue dans la structure client nommée "nblocation", qui répertorie chaque location effectuée par le client. Ceci permet à la fonction "appliquefidelite" de savoir à quelle échelle de fidélité se situe le client sélectionné lors de la location.Le programme vérifie et applique automatiquement la réduction appropriée à chaque client

2.4.Louer un véhicule

2.4.1 Algorithme de location

La fonction saisie_location qui permet de louer un véhicule a été la fonction la plus difficile à coder car c'est le cœur de notre sujet.
Après l'analyse des besoin métiers nous nous sommes aperçu que la fonction de location présente plusieurs défis lors de sa programmation :

1. Gestion des dates: La saisie et la validation des dates de début et de fin de location sont cruciales. Il faut s'assurer que les dates sont valides et qu'elles respectent certaines contraintes, comme ne pas être antérieures à la date actuelle.
2. Interaction avec l'utilisateur: Cette fonction nécessite une interaction régulière avec l'utilisateur pour saisir les informations sur le véhicule et le client. Il faut gérer les différentes possibilités de saisie, tout en offrant des messages d'erreur clairs en cas d'entrées incorrectes.
3. Vérification de la disponibilité des véhicules: Avant de valider une location, il est essentiel de vérifier la disponibilité du véhicule pour les dates sélectionnées. Cela implique de parcourir les locations existantes et de comparer les dates pour chaque véhicule.
4. Calcul des coûts de location: Le calcul du prix de la location peut être complexe, en particulier avec la prise en compte d'options supplémentaires, d'assurances et de réductions éventuelles.
5. Gestion des structures de données: La fonction manipule plusieurs structures de données, telles que les véhicules, les locations et les clients. Il est crucial de s'assurer que les données sont correctement stockées et mises à jour.

Pour aborder ces défis, nous avons suivi une approche systématique :

- Nous avons commencé par décomposer le processus de saisie de location en étapes distinctes, comme la saisie des dates, la sélection du véhicule et la saisie des informations du client.
- Nous avons conçu des fonctions modulaires pour gérer chaque aspect, ce qui favorise la réutilisation du code et facilite la maintenance.
- Nous avons utilisé des structures de données appropriées pour stocker les informations sur les véhicules, les clients et les locations, ce qui permet une manipulation efficace des données.

- Nous avons implémenté des mécanismes de validation robustes pour garantir que les données saisies sont correctes et cohérentes.
- Nous avons testé chaque fonctionnalité de manière approfondie, en prenant en compte différents scénarios d'utilisation et en nous assurant que le programme réagit de manière appropriée à chaque cas.

La structure location se compose donc des éléments suivants:

numero_loc: Un identifiant unique pour chaque location.
marque: La marque du véhicule loué. (Provient de la structure véhicule)
modele: Le modèle du véhicule loué. (Provient de la structure véhicule)
date_debut: La date de début de la location.
date_fin: La date de fin de la location.
nom_assurance: Le nom de l'assurance liée à la location, si une assurance est souscrite par le client.
assurance: Le coût de l'assurance par jour et TTC.
option: Le coût des options supplémentaires par jour et TTC.
prix: Le prix de la location (sans assurance ni options)par jour.
total: Le prix total de la location (incluant assurance et options).
caution: Le montant de la caution.
nom: Le nom du client louant le véhicule.(Provient de la structure client)
prenom: Le prénom du client louant le véhicule.(Provient de la structure client)
immatriculation: Le numéro d'immatriculation du véhicule loué.(Provient de la structure véhicule)
statut: Le statut de la location,"En cours" si le véhicule n’a pas été restitué et "Terminé" si le véhicule a bien été restitué par le client.
num_client: L'identifiant unique du client.(Provient de la structure client)
num_vehicule: L'identifiant unique du véhicule.(Provient de la structure véhicule)

Cette structure est bien organisée pour stocker toutes les informations nécessaires pour suivre les locations de véhicules, y compris les détails sur le véhicule lui-même, les dates de location, les informations du client et les coûts associés. De plus, le fait que cette structure comporte les clefs étrangères issues des structures client et véhicule nous permet de garantir l’intégrité, la cohésion et la qualité des données de façon automatique. Ainsi, nous pouvons retrouver l’intégralité des données d’un véhicule ou d’un client grâce à cette méthode à partir de la structure location.

En suivant cette approche, nous avons donc modélisé les besoins des métiers en réalisant un algorithme en pseudo-code. Cet algorithme vise à modéliser la future fonction de location et les fonctions qui lui sont liées.

```
saisie_location():
   Variable loue de type struct location
   Variable plaque, numero, ok, i, jeune_permis, days, stock, dispo de type entier
   Variable jour, mois, annee, jourr, moisr, anneer de type entier
   Variable date_valide_location, date_valider_location de type entier
   Variable decimal de type réel
   Variable choix_affichage de type tableau de caractères

   date_valide_location <- 0
   date_valider_location <- 0

   tant que date_valide_location est égal à 0 ou date_valider_location est égal à 0 faire
      Ecrire "Entrez la date de début de location souhaitée (JJ MM AAAA) : "
      Lire jour, mois, annee
      date_valide_location <- DateValideLocation(jour, mois, annee)
      Ecrire "Entrez la date de fin de location souhaitée (JJ MM AAAA) : "
      Lire jourr, moisr, anneer
      date_valider_location <- DateValideLocation(jourr, moisr, anneer)
      si date_valide_location = 0 ou date_valider_location = 0 alors
         Ecrire "Date invalide. Veuillez entrer des date valides."
      fin si
   fin tant que

   loue.date_debut <- format_date(jour, mois, annee)
   loue.date_fin <- format_date(jourr, moisr, anneer)
   days <- differenceJours(loue.date_debut, loue.date_fin)

   tant que vrai
      Ecrire "Pour vous aider, voulez-vous afficher les véhicules disponibles à la location pour ces dates ? (Oui/Non) : "
      Lire choix_affichage
      conv_maj(choix_affichage)
      si choix_affichage = "OUI" ou choix_affichage = "O" alors
         affichage_vehicule_dispo_date(loue.date_debut, loue.date_fin)
         sortir de la boucle
      sinon si choix_affichage = "NON" ou choix_affichage = "N" alors
         sortir de la boucle
      sinon
         Ecrire "Choix invalide, veuillez répondre par 'oui' ou 'non'"
      fin si
   fin tant que

   tant que vrai ou ok != 1 faire
      ok = 0
      Ecrire "Entrez le numéro du véhicule ('0' pour terminer) : "
      Lire plaque

      si plaque = 0 alors
         sortir de la boucle
      fin si
```

```
v <- recherche_numvehicule(plaque)

si v != -1 alors
   voiture <- tabvehicule[v]
   dispo <- VehiculeDispo(loue.date_debut, loue.date_fin, v)

   si dispo = 0 alors
      Ecrire "Le véhicule n'est pas disponible pour la location pour ces dates."
      sortir de la fonction de saisie de location
   fin si

   tabvehicule[v].loue <- "OUI"
   loue.num_vehicule <- voiture.num_vehicule
   loue.marque <- voiture.marque
   loue.modele <- voiture.modele
   loue.immatriculation <- voiture.immatriculation
   loue.caution <- voiture.caution

   Ecrire "Entrez le numéro client : "
   Lire numero

   n <- recherche_numclient(numero)

   si n != -1 alors
      individu <- tabclient[n]
      loue.num_client <- individu.num_client
      loue.nom <- individu.nom
      loue.prenom <- individu.prenom
      jeune_permis <- 2024 - individu.annee_permis

      si jeune_permis < 3 alors
         decimal <- voiture.surtaxe_jeunepermis / 100.0
         loue.prix <- voiture.prix * (1 + decimal)
      sinon
         loue.prix <- tabvehicule[v].prix
      fin si

      ok <- 1
      i <- nblocation
      loue.numero_loc <- i + 1
      loue.statut <- "EN COURS"
      tabloc[i] <- loue

      prix_assurance <- saisie_assurance()
      prix_option <- saisie_option()

      si prix_assurance > 0 alors
         afficher "Le prix de l'assurance sélectionnée est :"  prix_assurance
      fin si

      si prix_option > 0 alors
         afficher "Le prix de l'option sélectionnée est : " prix_option
      fin si

      loue.assurance <- prix_assurance
```

```
        loue.option <- prix_option
        loue.total <- (loue.assurance + loue.option + loue.prix) * days
        stock <- loue.total
        appliquerFidelite(loue, individu)
        individu.nblocation += 1
        a_sauvegarder <- 1
        tabclient[n] <- individu
        tabloc[nblocation] <- loue
        a_sauvegarderl <- 1
        si ok = 1 alors
          sortir de la boucle
        fin si
      sinon
        afficher "Numéro client non trouvé."
      fin si
    sinon
      afficher "Numéro du véhicule non trouvé."
    fin si
  fin tant que

  si ok = 1 alors
    reduction <- stock - loue.total
    Ecrire "Récapitulatif de la location :"
  fin si
  nblocation++
```

Fonctions appelés par saisie_location:

Fonction VehiculeDispo :

Cette fonction vérifie la disponibilité d'un véhicule pour une période donnée.
Elle parcourt les locations existantes et compare les dates pour déterminer si le véhicule est disponible ou non.
Cela garantit que les véhicules ne sont pas doublement loués pour les mêmes dates.

Fonction affichage_vehicule_dispo_date

Cette fonction affiche les véhicules disponibles pour une période donnée.
Elle parcourt les véhicules et vérifie leur disponibilité en fonction de leurs statuts et des périodes de location prévues.
Les véhicules disponibles et non loués sont affichés directement, tandis que ceux qui sont loués pour la période spécifiée ne sont pas affichés.

Fonction crefacture () :

Cette fonction crée une facture pour une location donnée. Elle génère un numéro de facture aléatoire, copie les informations de la location vers la facture, puis ajoute la facture à un tableau de factures.

Fonction estBissextile_:
Cette fonction vérifie si une année donnée est bissextile ou non. Elle est utilisée pour déterminer le nombre de jours dans le mois de février.

Fonction DateValideLocation_:
Cette fonction vérifie si une date de location est valide. Elle compare la date de location avec la date actuelle pour s'assurer qu'elle est postérieure ou égale à la date actuelle.

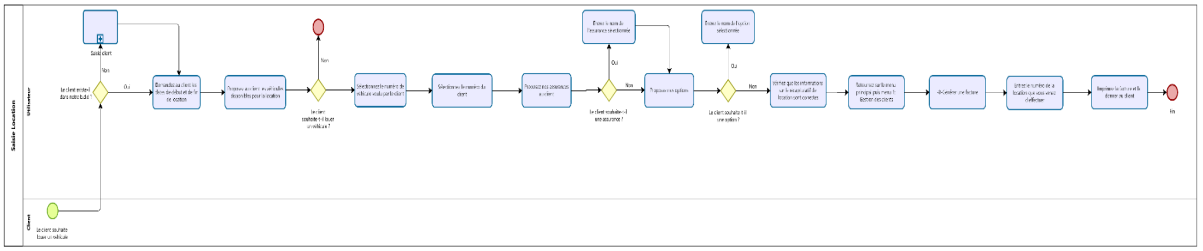
Fonction format_date_:
Cette fonction formate une date au format "jj/mm/aaaa" et retourne une chaîne de caractères contenant la date formatée.

Fonction differenceJours_:
Cette fonction calcule la différence en jours entre deux dates données. Elle convertit les dates en time_t, puis calcule la différence en secondes avant de la convertir en jours.

2.4.2 Processus utilisateur

Afin de faciliter la saisie d'une location par les utilisateurs nous avons créé un processus au norme BPMN, c'est un processus visuel qui représente les étapes de notre processus de location de manière claire et compréhensible.

Pour une meilleure visibilité nous vous avons transmis le processus suivant dans le dossier en pièce-jointe.



2.6 factures et suivie des locations

Les factures sont essentielles dans chaque entreprise pour s'assurer un suivi des flux monétaire sortant et en occurrence entrant.

La structure Facture possède les mêmes caractéristiques que la structure de location. Néanmoins la structure facture possède quelques particularités, comme son propre ID nommé num_facture automatiquement incrémenté qui répertorie les numéro de factures et la date_facture qui enregistre la date à laquelle la facture est faite.

2.6.1 Automatisations et historique des location

Pour chaque location entrée, une facture est créée et est automatiquement entrée dans l'historique des factures. A défaut de posséder un historique des locations, l'historique des factures permet de pouvoir conserver l'historique de chaque location de façon plus synthétique et condensée.

2.6.2 Affichage facture

L'affichage des factures permet d'obtenir une vue globale sur l'intégralité des données du tableau "tabfacture". Ceci permettant d'une part d'accéder à l'historique des factures et par ce fait des locations. Cela permet de visualiser les flux d'argent entrant, généré par l'activité de location de voiture de l'agence.

2.6.3 Générer une facture client

L'application propose la génération d'une facture sur une location particulière. Contraire à la vue macro proposée par l'affichage, la génération de facture client elle propose une vue micro bien plus précise et détaillée sur une location particulière. Cette fonctionnalité vise à satisfaire la demande d'un client de recevoir une facture de sa location.

2.6.4 Reporting de l’activité

L’application propose de faire un reporting mensuel de l’activité de l’entreprise. Ce qui permet à l’utilisateur d’avoir une meilleure visibilité sur le profit généré par son activité, tout en prenant en compte le nombre de location faîte chaque mois.

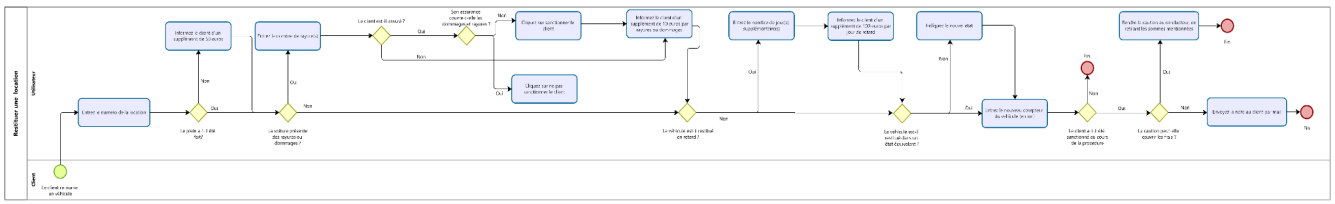
2.7 Restituer un véhicule

La fonction retour_vehicule gère le processus de restitution d'un véhicule loué. Elle vérifie si une location correspondant au numéro donné est en cours, puis guide l'utilisateur à travers différentes étapes telles que la vérification du plein d'essence, des dommages éventuels, du retard de retour, de l'état du véhicule, et du kilométrage. En fonction des réponses de l'utilisateur, elle calcule d'éventuels suppléments à appliquer sur la caution du client et met à jour les informations de la location et du véhicule.

La restitution d’un véhicule est un moment crucial pour les boites de location de véhicules. En effet, il faut aller vite car le client est souvent pressé (vol,train à prendre…) mais tout en constatant de façon pertinente et juste si le client a respecté les règles du contrat de location (exemple: le plein doit être fait entièrement) et si le véhicule n’a pas subi des dommages supplémentaires. Pour faciliter la restitution d’un véhicule nous proposons un processus simple et clair pour l’utilisateur. Nous procédons de la façon suivante:

Tout d'abord, nous vérifions si le plein d'essence a été fait. Si ce n'est pas le cas, nous ajoutons un supplément de 50 euros. Ensuite, nous évaluons les dommages éventuels, tels que les rayures sur la carrosserie. Si le client possède une assurance adaptée, le client est exempté. Autrement, un supplément de 10 euros par rayure est retiré de la caution. De plus, en cas de retard dans la restitution du véhicule, un supplément de 100 euros par jour de retard est appliqué. Enfin, si l'état du véhicule est différent de celui au moment de la location, une mise à jour de l'état est effectuée. Tous ces suppléments sont pris en compte pour calculer le montant final à déduire de la caution du client ou à payer en sus.
Tout ce processus permet d'assurer la bonne gestion des coûts associés à la restitution du véhicule et de faciliter le processus de restitution pour les boites de location.

Nous avons également réalisé le processus suivant, processus qui doit être suivi par toutes les équipes chargées de la restitution des véhicules. Pour une meilleure visibilité du processus, il est disponible dans le dossier au format png.



Les atouts du programme

En analysant les besoins métiers et leurs demandes, nous avons décidé de mettre en place des fonctionnalités supplémentaires afin d'aider les métiers à mieux gérer les locations de véhicules.

3.1 Assurances, options, gestion de la fidélité et gestion des factures

Les atouts de notre programme incluent une gestion exhaustive des besoins associés à la location de véhicules. Le programme propose de gérer efficacement le processus de fidélité, les assurances, options et les factures liés à la location de véhicules, offrant ainsi une solution complète.

Les options et les assurances sont des éléments cruciaux pour une boite de location de véhicule car ce sont majoritairement en vendant ce genre de service que l'entreprise fait de la marge.
Notre programme propose trois types d'assurances mais il est possible d'en ajouter d'autres si la boîte de location évolue. Tout d'abord, nous avons l'assurance tout risque - 25.00€ : Cette assurance offre une couverture complète en cas d'accident, de dommages au véhicule ou de vol. Une assurance contre les rayures - 10.00€ : Cette assurance spécifique couvre les frais de réparation en cas de rayures ou d'éraflures sur le véhicule pendant la période de location. Pour terminer, nous avons une assurance contre le vol - 20.00€ : Cette assurance protège le client en cas de vol du véhicule pendant la période de location.

Nous proposons également trois options mais il est possible d'en ajouter d'autres si la boîte de location évolue. Tout d'abord, nous avons l'option GPS qui coûte 10.00€ : Cette option permet d'équiper le véhicule d'un GPS. Une option siège-bébé qui coûte 5.00€ : Cette option permet d'équiper le véhicule avec un siège-bébé. L'option conducteur supplémentaire coûte 15.00€ : Cette option permet au conducteur de ne pas être le seul à pouvoir utiliser le véhicule.

3.2 Historicisation des données

L’Historisation des données est un réel avantage pour les entreprises, surtout dans un secteur où les informations client et véhicule sont si critiques. Cette fonction permet de conserver des archives des transactions, des locations, et des interactions clients, même après leur suppression active du système. Cela aide non seulement à respecter les réglementations sur la protection des données, comme le RGPD, mais offre aussi la possibilité de réactiver rapidement les informations des clients si nécessaire.

3.4 Adaptabilité et évolutivité

L'adaptabilité et l'évolutivité sont des caractéristiques essentielles de notre programme. Conçu pour s'adapter aux besoins changeants des entreprises de location, le système peut facilement intégrer de nouvelles fonctionnalités, options, ou ajustements selon les exigences du marché ou de la réglementation. Cette flexibilité assure que l'application reste pertinente et utile même à mesure que l'entreprise grandit ou évolue.

3.5 Optimisation de la gestion des retours et des dommages

Enfin, notre programme offre une gestion optimisée des retours de véhicules et des évaluations des dommages. Cela inclut des outils automatisés pour inspecter et documenter l'état des véhicules au moment de leur retour, facilitant ainsi le processus de facturation des dommages si nécessaire. Cette approche minimise les conflits et les malentendus avec les clients, tout en protégeant les actifs de l'entreprise. A la suite d’une série de questions visant à enregistrer le nouvel état du véhicule, l’application calcule le coût des frais en cas de dégâts. Ce dit coûts est directement répercuté sur la caution et met à jour l’état du véhicule.

3.6 Comptabilité

L’application propose un service de comptabilité. Ce service propose de calculer le chiffre d'affaires des 30 derniers jours de l’agence. De plus, l’activité de l’agence y est reportée, ainsi le propriétaire est en capacité de savoir combien de location ont été faites tout au long du mois. Ce service permet de mettre en place un reporting simple mais efficace de l’activité de l’agence.

3.7 Mettre en place des contrôles de saisie performants

3.7.1 Garantir des données de qualité

La qualité des données est aujourd'hui un point extrêmement important pour les entreprises. Des données de qualités permettent de…
C'est pour ces raisons que nous avons décidé de mettre en place des contrôles de saisie dans notre programme afin
Il faut trouver un juste milieu car si on met en place énormément de contrôle de saisie, le programme devient inutilisable pour l'utilisateur.

3.7.2 Exemple de contrôle de saisie

Exemple de la fonction de saisie client. Avoir des données de qualité sur ses clients est obligatoire au bon fonctionnement d'une entreprise de location.
Avoir un numéro de téléphone et une adresse mail, une adresse valide permet la prospection, l'envoi de promotion par l'équipe marketing.
Il y a également un volet juridique quand on loue une voiture on doit savoir à qui on la loue et si la personne dispose de l'âge nécessaire à la location et d'un permis de conduire valide. Ces informations sont donc essentielles.

1. Contrôle de la date de naissance :
    - Le programme demande à l'utilisateur de saisir la date de naissance du client au format DD MM YYYY.
    - Il vérifie ensuite si la date est valide à l'aide de la fonction DateValide. Si la date n'est pas valide, l'utilisateur est invité à saisir à nouveau une date valide.
2. Contrôle du numéro de permis :
    - L'utilisateur est invité à saisir le numéro de permis du client, qui doit être composé de 12 caractères.
    - Si la longueur du numéro de permis n'est pas égale à 12, l'utilisateur est invité à saisir à nouveau un numéro de permis valide.
3. Contrôle du numéro de téléphone :
    - L'utilisateur est invité à saisir le numéro de téléphone du client, qui doit être composé de 10 chiffres et commencer par 0.
    - Il vérifie également que le numéro ne contient pas seulement des zéros dans les chiffres intermédiaires.

- Si le numéro de téléphone ne respecte pas ces conditions, l'utilisateur est invité à saisir à nouveau un numéro de téléphone valide.

4. Contrôle du mail :
    - L'utilisateur est invité à saisir l'adresse e-mail du client.
    - Le programme vérifie que l'adresse e-mail contient au moins un "@" et au moins un "." et qu'elle contient plus de 5 caractères.
    - Si ces conditions ne sont pas remplies, l'utilisateur est invité à saisir à nouveau une adresse e-mail valide.
5. Contrôle du type de voie :
    - L'utilisateur est invité à saisir le type de voie (rue, avenue, etc.).
    - Le programme vérifie si le type de voie saisi est valide en le comparant à une liste prédéfinie.
    - Si le type de voie n'est pas valide, l'utilisateur est invité à saisir à nouveau un type de voie valide.

3.7.3 Gestion des dates

En analysant les besoins métiers nous avons observé que ce qui est le plus difficile à gérer dans une agence de location c'est les dates. Imaginons que deux clients se présentent le même jour pour récupérer le même véhicule et c'est le drame. Pour éviter ce désagrément aux équipes nous avons automatisé la gestion des dates dans le programme.

En utilisant notre programme, une location ne peut en aucun cas s'effectuer dans le passé. Il est également possible pour les clients de réserver le même véhicule mais il faut impérativement que les dates soient différentes et ne se chevauchent pas.

Bilan critique

Le développement de notre application de gestion de location de véhicules a été une expérience enrichissante et instructive. Toutefois, comme tout projet, il comporte ses réussites et ses défis. Ce bilan critique vise à évaluer les choix effectués, à identifier les difficultés rencontrées et à envisager des améliorations futures.

Réussites du projet

1. Adaptabilité et évolutivité : Le système a été conçu pour être flexible, permettant une adaptation facile aux besoins changeants des entreprises de location. Cela inclut la possibilité d'ajouter de nouvelles fonctionnalités ou d'adapter les existantes sans perturber le fonctionnement général.

2. Interface utilisateur intuitive : L'interface de l'application a été développée pour être facile à utiliser, permettant aux utilisateurs de tous niveaux techniques de naviguer et d'utiliser les fonctionnalités sans difficulté.

3. Automatisation: Le système automatise certaines tâches, telle que la création de factures, l'application de la réduction de la fidélité etc…
   Une automatisation nécessaire pour faciliter l'utilisation de l'application et soulager au quotidien les propriétaires d'agences de location.

Défis rencontrés

1. Complexité de la gestion des dates : La gestion des dates de location a été particulièrement complexe, surtout en ce qui concerne la validation des périodes de location pour éviter les chevauchements. Cela a nécessité un développement spécifique pour s'assurer que les données soient cohérentes et que les erreurs soient minimisées.

2. Gestion des données en format texte : Bien que le choix d'une base de données en format texte ait simplifié certains aspects du développement, il a également limité les fonctionnalités que l'on pourrait attendre d'un système de gestion de base de données plus robuste, comme les requêtes complexes et la manipulation efficace des données.

Possibilités d'amélioration

1. Migration vers une base de données relationnelle : Pour surmonter les limitations d'une gestion de données basée sur des fichiers texte, il serait bénéfique de migrer vers un système de gestion de base de données relationnelles. Cela améliorerait la sécurité, la fiabilité et la flexibilité de la gestion des données.

2. Amélioration de l'interface utilisateur : Bien que l'interface actuelle soit fonctionnelle, elle pourrait être améliorée avec des technologies plus modernes pour offrir une expérience utilisateur plus riche et plus interactive.

3. Extension des fonctionnalités de reporting : Développer davantage les outils de reporting pour fournir des analyses plus détaillées et des visualisations des données, ce qui aiderait les gérants à mieux comprendre leur activité et à prendre des décisions basées sur des données.

En conclusion, bien que notre application réponde aux besoins actuels des entreprises de location de véhicules, il est certain que des améliorations continuelles seront nécessaires pour s'adapter aux évolutions du marché et aux exigences technologiques. La prise en compte des retours des utilisateurs et l'anticipation des besoins futurs seront essentielles pour le succès continu du projet.