Cahier des charges

PRÉSENTATION GÉNÉRALE

CONTEXTE

La SNCF est une compagnie ferroviaire ayant entre autres activités le transport de voyageurs. Elle souhaite disposer d'un logiciel de réservation de place : `reservation`.

PORTÉE

`reservation` est un programme de réservation de place dans les différents modes de transport assurés par la SNCF. Ce prototype est limité à la réservation de places :

- de TGV uniquement
- pour un échantillon des données de trajets
- à horizon d'1 mois pour la réservation
- à 45 places par rame

Il permet 6 fonctions principales :

- la consultation d'horaires de voyage entre deux gares données à une date donnée
- la consultation des tarifs de ces voyages selon la classe du wagon
- la réservation de places
- la gestion des réservations d'un client
- la gestion des places disponibles à bord d'un train
- le calcul des tarifs de voyage

`reservation` ne permet pas :

- la réservation de voyages incluant des correspondances.

DÉFINITIONS

billet, **billet de train** : titre de transport valant pour une place et pour un voyage.

classe : catégorie de prestation offerte à bord d'un wagon. Certains trains possèdent des wagons de différentes classes : première classe (classe 1) et seconde classe (classe 2). D'autres trains possèdent des wagons d'une seule et même classe.

date système, **date/heure système** et **heure système** : informations de date, date/heure et heure de l'équipement terminal sur lequel le programme est utilisé.
D'un point de vue métier, il s'agit de la date, date/heure et heure courantes, c'est-à-dire la date, date/heure et heure d'utilisation du programme, sachant que le comportement du

programme dépend de cette date, date/heure et heure (les voyages proposés ont une date/heure postérieure à la date courante).
Dans le programme, ces informations sont obtenues via l'équipement terminal sur lequel le programme est exécuté. Elles s'appellent donc date système, date/heure système et heure système malgré la consonance très peu « métier ».

- **date** désigne le jour calendaire d'une année donnée (jour de semaine, jour du mois, mois, année). *Exemple : mercredi 17/02/2021*.
- **heure** désigne un horaire (heures, minutes et secondes). *Exemple : 23:59:00*.
- **date/heure** désigne un horaire d'une date. *Exemple : mercredi 17/02/2021 23:59:00*.

gare : station à partir ou au départ de laquelle il est possible de faire un voyage assuré par la SNCF. Il peut s'agir d'une gare de chemin de fer ou d'une gare routière. Une ville peut posséder plusieurs gares.

place : siège à bord d'un train pour une séquence d'un voyage, réservable par un voyageur.

rame : train au sens « matériel roulant», caractérisée par une composition en siège.

terminal : équipement terminal, appareil, sur lequel le programme est utilisé. Dans la version courante du programme, seul l'ordinateur peut être un terminal.

train : catégorie mixte de véhicule / service commercial, dont la circulation est assurée par la SNCF et dont l'utilisateur peut réserver une place. Les trains sont de différents types, que la rame circule sur voie ferrée ou routière (TGV, INOUI, TER, OUIGO, Car).

trajet : combinaison unique de gares desservies par un train, et d'horaires d'arrivée et de départ dans chaque gare.

séquence : portion de trajet entre deux gares d'arrêt consécutives.

service : combinaison d'un trajet et d'un calendrier de circulation (circule ou ne circule pas, pour chaque jour de semaine).

siège : unité de composition d'une rame où un voyageur peut s'asseoir.

voyage : déplacement en train entre deux gares pour un trajet et une date donnée.

VUE D'ENSEMBLE

La suite du document est divisée en trois parties : une analyse métier de la réservation de places et une description des spécifications du programme, une analyse des limites du programme.

ANALYSE MÉTIER

La SNCF assure la circulation de train de voyageurs. La gestion de la réservation doit tenir compte des caractéristiques des trains, des informations de circulation des trains, des informations de disponibilité des places, des conditions tarifaires et d'informations concernant le voyageur.

UTILISATEUR

L'utilisateur peut être un agent ou un voyageur.

CIRCULATION DES TRAINS

Les trains circulent en suivant des trajets, découpés en séquences caractérisées par une gare et une heure de départ et une gare et une heure d'arrivée, consécutives sur le trajet, selon un calendrier de service.

Trajet

Un train circule en suivant un trajet. Un trajet est une combinaison unique de gares desservies par un train, et d'horaires d'arrivée et de départ dans chaque gare.

Un trajet peut être décrit comme suit :

> Un trajet effectuant la liaison depuis Bordeaux jusqu'à Nice et s'arrêtant à Marseille, dans cet ordre, partant de Bordeaux à 14h03 et arrivant à Nice à 18h47.

Service / Calendrier

Un trajet est associé à un service qui définit le calendrier hebdomadaire de circulation, et des dates de début et fin de validité.

Un service peut être décrit comme suit :

> Le service concerne tous les mardis et tous les jeudis du 27 Novembre 2020 au 3 Mars 2021.

Un trajet est donc effectué, toujours suivant les mêmes horaires de départ et d'arrivée, certain(s) jour(s) de la semaine, d'une date de début à une date de fin de validité.

Le trajet peut alors être décrit comme suit :

> Un trajet effectuant la liaison depuis Bordeaux jusqu'à Nice et s'arrêtant à Marseille, dans cet ordre, partant de Bordeaux à 14h03 et arrivant à Nice à 18h47, tous les mardis et tous les jeudis du 27 Novembre 2020 au 3 Mars 2021.

Séquence

Une séquence est un élément constitutif d'un trajet, ce dernier étant composé d'une suite de séquences continues et cohérentes.

Une séquence est caractérisée par une gare de départ, une heure de départ, une gare d'arrivée et une heure d'arrivée. Pour un trajet, il y a donc une séquence de moins qu'il n'y a de gares.

Un numéro de séquence est égal au numéro de « stop » de départ de cette séquence.

Une séquence peut donc être décrite comme suit :

> La séquence numéro 3 commence à la troisième gare du trajet et finit à la quatrième gare du trajet.

Gare

Une gare est une station où s'arrête un train. Elle est identifiée par un identifiant ou un nom.

Une gare possède des zones d'embarquement, identifiées par un identifiant, et le nom et l'identifiant de la gare à laquelle elle appartient.

Une gare peut être incluse dans plusieurs trajets. Dans un trajet, une gare représente un arrêt, identifié par un numéro d'arrêt. Une gare peut alors être identifiée par le numéro d'arrêt dans le trajet.

Une gare est incluse dans deux séquences consécutives d'un même trajet : elle est la gare de départ d'une séquence X et la gare d'arrivée de la séquence X-1.

Une gare peut être décrite comme suit :

> son nom : Gare de Lyon-Part-Dieu
>
> son identifiant : StopArea:OCE87723197
>
> l'identifiant d'un de ses points d'arrêt : StopPoint:OCETGV INOUI-87723197
>
> son numéro d'arrêt dans un trajet donné : n°7 du trajet OCESN009877F58058-1323290491
>
> la gare d'arrivée de la séquence 6 du trajet OCESN009877F58058-1323290491
>
> la gare de départ de la séquence 7 du trajet OCESN009877F58058-1323290491

OUVERTURE DES VENTES

Les ventes de billets sont ouvertes avec une antécédence, par rapport au voyage, différente selon le type de train :

- TGV, INOUI & Intercités : 4 mois, sauf pour les vacances d'été, de Noël et d'hiver et de printemps. L'ouverture des ventes des voyages des vacances est plus tardive que les 4 mois habituels.

- TGV Bruxelles, Italo, TGV Luxembourg-Paris, TGV Fribourg-Paris : 4 mois
- Trajets RENFE/SNCF, ALLEO France-Allemagne, TGV Lyria (entre Lille et Genève) : 4 mois
- TGV France-Italie : 4 mois
- TER : en fonction des régions, entre 2 et 5 mois
- OUIGO : entre 2 et 9 mois
- THALYS (vers la Belgique, les Pays-Bas, l'Allemagne) : 4 mois
- Eurostar (vers l'Angleterre) : 6 mois pour Paris, Lille et Calais vers London, Ashford, Ebbsfleet, 150 à 280 jours pour Lyon, Marseille et Avignon, vers London, Ashford, Ebbsfleet

Lorsqu'un voyage est ouvert à la vente, ses places doivent être réservables.

TRAIN

La SNCF fait circuler différents types de trains, indiqués par le type TGV, INOUI, OUIGO, TER, Intercités, Car. Le « type » désigne des réalités mixtes, à la fois commerciales et techniques.

Le train au sens de « matériel roulant » est désigné par « rame ».

Rame

Un type de train peut utiliser un ou plusieurs types de rames. Un train de type Car utilisera probablement une rame standard (un car) avec un nombre de places fixe, tandis qu'un train de type TGV utilise différents types de rames, selon la ligne de chemin de fer (une rame duplex sur Paris-Lyon, une rame simplex sur une autre ligne ou sur un autre service.

Chaque type de rame a une composition propre, mais de façon commune :

- une rame est constituée d'un ou plusieurs wagons
- les wagons d'une rame sont de même classe ou de classes différentes
- un wagon est simplex ou duplex, il possède alors une ou deux salles (salle basse et salle haute)
- une salle est constituée de sièges numérotés
- un siège a un placement et un numéro. Selon le type de rame et la classe de wagon dans laquelle se trouve le siège, le placement peut être : fenêtre, couloir, isolé.

L'information du type de rame utilisée en fonction du trajet est difficile à trouver. L'affectation des rames aux trajets semble avoir une base en partie régionale, liée aux commandes de rames lors de l'ouverture d'une nouvelle ligne. À titre d'exemple, un TGV peut utiliser une rame TGV Sud-Est, TGV Atlantique, TGV Réseau, TGV Duplex, TGV Réseau Duplex, TGV Duplex Dasye, TGV 2N2, TGV PBA, TGV TMST, TGV POS, TGV 2N2, TGV IRIS 320, TGV M, TGV PBKA, Eurostar e320 Wikipédia : Matériel moteur de la SNCF # Parc électrique.

Dans le cadre de ce prototype, nous utilisons une composition de rame identique et échantillonnée pour tous les services d'un même type de train, tout en permettant un paramétrage différent au cas où ces informations deviennent accessibles.

Wagon

Un wagon possède les caractéristiques suivantes :

- Un numéro
- Une classe : unique, première classe ou seconde classe. La classe influe sur le tarif.
- Un nombre de salle : 1 ou 2.

Salle

Une salle possède les caractéristiques suivantes :

- Un type de salle : unique, haute, basse

Une salle contient des sièges disposés de façon irrégulière.

Siège

Un siège possède les caractéristiques suivantes :

- Une position : fenêtre, couloir ou isolé
- Un numéro.

Une Place

Une place est un siège à bord d'une rame pour une séquence donnée d'un voyage donné (trajet et date de circulation).

Une place est identifiée par le numéro de wagon et le numéro de siège. Elle est aussi caractérisée par sa salle, son placement, et son état de réservation (libre ou occupée).

Une place est réservable lorsque le voyage auquel elle correspond est ouvert à la vente et que la place est libre. Lorsque le voyage comporte plusieurs séquences, les places de mêmes identifiants (wagon, siège) de chaque séquence doivent être libres pour que la réservation soit possible.

Composition des rames

La liste des sièges d'une rame en fonction des wagons, salle et position, n'est pas régulière. La hiérarchie « wagon, salle, siège » ne peut pas être conservée. Un plan de rame est utilisé pour caractériser chacun de ses sièges : une rame est décrite par la liste des sièges et ses caractéristiques (wagon, classe, salle, position, numéro).

TGV, INOUI

Une rame de train de type TGV ou INOUI est composée de 8 wagons :

- 7 wagons de passagers (n°1 à 3 et n°5 à 8)
- 1 wagon-bar (n°4)

Parmi les 7 wagons de passagers, 3 sont de classe 1 (n°1 à 3), 4 sont de classe 2 (n°5 à 8).

Une rame de train de type TGV ou INOUI peut être simplex ou duplex.

Une salle de wagon de classe 1 propose 3 placements de sièges : fenêtre, couloir, isolé

Une salle de wagon de classe 2 propose 3 placements de sièges : fenêtre, couloir

Rame simplex

Une rame simplex est composée de wagons à 1 niveau (une salle).

Chaque génération de rame possède son propre plan de placement :

- Placement des rames TGV Sud-Est Rénové 1 (351 places)

Rame duplex

Une rame duplex est composée de wagons à 2 niveaux, et possède ainsi une salle basse et une salle haute.

Une salle de wagon de classe 1 propose 3 placements de sièges : * fenêtre * couloir * place isolé

Une salle de wagon de classe 2 propose 3 placements de sièges : * fenêtre * couloir

Chaque génération de rame possède son propre plan de placement :

- Placement des rames TGV DUPLEX GÉNÉRATION 1 (510 places) (510 places)
- Placement des rames TGV DUPLEX GÉNÉRATION 3 «EURO DUPLEX » (509 places)

Autres Modes de Transport Gérés par la SNCF

La SNCF gère plusieurs autres modes de transport :

- Des trains :
  - OUIGO
  - TER
  - Intercités
- Des Cars :
  - Car
  - Navette
- Des tramways

Ces modes de transport n'ont pas les mêmes caractéristiques que les TGV. La gestion de ces modes de transport a un impact sur les caractéristiques suivantes :

- La classe : possiblement une classe unique au lieu de 2.
- Le nombre de salles : possiblement réduit à une.
- Le nombre de sièges.
- La position des sièges : possiblement moins de positions différentes.

TARIFS

Le tarif dépend de la tarification et des programmes de réduction.

Tarification

La tarification dépend du type de train.

TGV

Trois tarifs sont proposés pour les voyages en TGV : prix d'appel 2nde, plein tarif loisir 2nde, 1re classe.

- Le prix d'appel 2nde est un tarif proposé à l'ouverture des ventes, pour un voyage en seconde classe. Il offre un niveau de prestation réduit concernant les conditions d'échange et d'annulation par rapport au plein tarif loisir 2nde. Il évolue en fonction du remplissage du train.
- Le plein tarif loisir 2nde est un tarif proposé pendant toute la période de réservation, pour un voyage en seconde classe.
- Le plein 1re classe est un tarif proposé pendant toute la période de réservation, pour un voyage en première classe.

Chaque tarif est défini pour une gare de départ et une gare d'arrivée.

Programmes de réduction

Deux types de programmes de réduction sont proposés.

- Des cartes de réduction.
- Des abonnements.

RÉSERVATION

Le processus de réservation d'une place de train suit généralement les étapes suivantes :

1. Recherche
   - choix de la gare/ville de départ
   - choix de la gare/ville d'arrivée

   - choix de la date
   - choix des options de voyage : classe de wagon, programme de réduction.
2. Consultation des résultats
3. Modification des critères de recherche
4. Choix et réservation d'un voyage parmi les résultats, avec renseignement des informations utilisateurs

Recherche

Un utilisateur doit pouvoir consulter les horaires de voyages en indiquant :

- la ville de départ
- la ville d'arrivée
- la date du voyage

Choix de la gare/ville de départ

Certaines villes possèdent plusieurs gares. C'est le cas :

- des grandes villes et des villes possédant une gare de chemin de fer historique et une gare TGV située en dehors de la ville sur une ligne à grande vitesse.
- des villes possédant une gare de chemin de fer et une gare routière

Le voyageur peut souhaiter réserver une place pour un voyage au départ ou à l'arrivée d'une ville, indépendamment de la gare de départ ou d'arrivée dans cette ville. Les résultats affichés doivent inclure les différentes gares d'une même ville.

Si aucun train ne circule au départ de cette ville, le processus de recherche doit s'interrompre. Il peut éventuellement proposer de saisir une autre ville de départ.

Choix de la gare/ville d'arrivée

La problématique gare/ville est identique pour la gare/ville d'arrivée.

Si aucun train ne circule entre la ville de départ et la ville d'arrivée, le processus de recherche doit s'interrompre. Il peut éventuellement proposer de saisir une autre gare/ville d'arrivée en conservant la gare/ville de départ précédemment choisie.

Choix de la date

La date de voyage ne peut pas être antérieure à la date système.

Si la date saisie est antérieure, le processus de recherche doit s'interrompre. Il peut éventuellement proposer la saisie d'une autre date en conservant la gare/ville de départ et la gare/ville d'arrivée précédemment choisies.

Gestion de la date/heure

Concernant la date/heure, le cas de la réservation de voyage et le cas de la consultation d'horaires sont différents.

Dans le cas d'une réservation de voyage, le programme ne doit pas proposer de résultats de voyages partant à une date/heure antérieure à la date/heure système.

Dans le cas d'une consultation d'horaire, le programme ne doit pas proposer de résultats de voyages arrivant à une date/heure antérieure à la date/heure système.

Consultation des résultats

Les résultats de la recherche doivent être affichés et des actions doivent être proposées à l'utilisateur.

Affichage des résultats

L'affichage des résultats doit faire apparaitre suffisamment d'informations pour permettre à l'utilisateur de choisir entre continuer le processus de réservation ou y apporter des modifications, sans qu'il y ait de surabondance d'informations.

Il est nécessaire d'afficher :

- La gare de départ
- La gare d'arrivée
- L'heure de départ
- L'heure d'arrivée
- Le type de train
- Le prix ou l'indisponibilité en première classe
- Le prix ou l'indisponibilité en seconde classe

Les résultats doivent être affichés par heure de départ croissant. Un autre critère de tri peut être proposé à l'utilisateur.

Proposition d'actions

Des résultats proposés doivent découler des actions.

Les actions possibles pour l'utilisateur à la vue des résultats sont :

- Réserver un des trajets proposés
- Modifier sa recherche
- Quitter le processus de recherche

Réserver

Si l'utilisateur est satisfait par un des résultats proposés, il doit pouvoir sélectionner l'un des trajets pour lancer le processus de réservation.

Modification des critères de recherche

Un utilisateur doit pouvoir modifier les critères suivants de sa recherche :

- la ville de départ
- la ville d'arrivée
- la date du voyage

La modification de la ville départ ou de la ville d'arrivée peut être assimilée au fait de quitter le processus de recherche et d'en recommencer un nouveau.

Nous ne traiterons donc que de la modification du paramètre date. Ce dernier peut être modifié de deux façons :

- Par incrémentation : le jour suivant.
- Par décrémentation : le jour précédent.

Réservation d'un voyage parmi les résultats

Après le choix d'un trajet, plusieurs options sont proposées à l'utilisateur concernant son voyage.

La disponibilité de la combinaison de ces préférences doit être vérifiée.

Lorsque les choix sont fixés, des informations concernant les voyageurs sont demandées à l'utilisateur.

La réservation passe alors en paiement.

Lorsque le paiement est effectué, les billets sont générés.

Choix Utilisateur

Après le choix d'un trajet, plusieurs options doivent être proposées à l'utilisateur concernant le choix des sièges. L'utilisateur doit être invité à renseigner ses préférences concernant :

- la classe du wagon
- le nombre de places : dans la limite du nombre de places disponibles.
- la salle : si le train possède plus d'une salle, proposer la salle basse ou la salle haute.
- la position de son siège :
  - fenêtre ou couloir s'il voyage en seconde classe
  - choix du siège parmi les sièges disponibles s'il voyage en première classe

Vérification

La disponibilité de la combinaison de ces préférences doit être vérifiée pour poursuivre le processus de réservation. Par exemple, il peut y avoir des sièges restant en première classe, mais aucun côté fenêtre. Les choix de l'utilisateur étant fait les uns après les autres, chaque

choix influe sur la faisabilité du choix suivant. La vérification doit être faite à chaque étape de choix.

Si la combinaison est disponible, le processus doit continuer. Si elle n'est pas disponible, l'utilisateur doit être notifié. Le choix doit lui être donné de poursuivre la réservation avec une combinaison de critères différente de ses préférences ou de modifier son choix de trajet.

S'il choisit de poursuivre, le processus de réservation passe au calcul du montant total, puis au paiement, puis à la création des billets et l'inscription de la réservation dans la base de données.

Classe du wagon

Les options suivantes dépendent de la classe choisie.

Nombre de places

Il est aussi possible que l'utilisateur souhaite réserver plusieurs sièges. Il faut demander à l'utilisateur de modifier son choix tant qu'il n'a pas renseigné un nombre de sièges désirés inférieur au nombre de sièges disponibles pour une classe donnée.

Seconde classe

Lorsque l'utilisateur précise qu'il veut un siège ou plusieurs sièges en seconde classe et que sa demande est faisable, il ne doit être laissé le choix à l'utilisateur de préciser sa demande que si plusieurs solutions sont possibles à chaque fois. Sinon, le choix des caractéristiques de salle et de position doit lui être imposé.

S'il ne reste des places disponibles que dans une salle, il faut prévenir l'utilisateur.

S'il reste des places disponibles dans les deux salles, il faut proposer à l'utilisateur de choisir.

S'il ne reste des places disponibles que pour une seule position de siège dans cette salle, il faut prévenir l'utilisateur.

S'il reste des places disponibles dans plusieurs positions de sièges dans cette salle, il faut proposer à l'utilisateur de choisir.

Enfin, il faut demander à l'utilisateur si ses choix, ou ces assignations, lui conviennent, ou s'il veut revenir au choix du trajet.

Première Classe

Lorsque l'utilisateur précise qu'il veut un siège ou plusieurs sièges en première classe et que sa demande est faisable, il faut lui présenter la liste des places disponibles en première classe, avec les caractéristiques de chaque place.

L'utilisateur renseigne alors son ou ses choix.

Une place ne pouvant être choisie deux fois, un choix n'est plus affiché par la suite.

Enfin, il faut demander à l'utilisateur si ses choix lui conviennent, ou s'il veut revenir au choix du trajet.

Paiement

Lorsque le processus de réservation continue, soit que la combinaison des préférences était disponible, soit que l'utilisateur a choisi de poursuivre la réservation avec une combinaison proposée, le processus passe à une étape de calcul du montant total. Le montant total de la réservation doit être présenté à l'utilisateur.

Le processus de paiement doit demander des informations de paiement à l'utilisateur :

- Le numéro de carte bancaire
- La date d'expiration de la carte bancaire
- Le code CVC de la carte bancaire

Le succès de la procédure de paiement doit être notifié à l'utilisateur.

Export de la réservation

Une fois le paiement effectué avec succès, la réservation est considérée faite. Les informations de réservation doivent être stockées dans la base de données.

Les places sélectionnées doivent être mentionnées comme étant désormais non disponibles sur l'intégralité des séquences sur lesquelles l'utilisateur souhaite voyager.

Création du billet

Pour terminer le processus de réservation, un billet récapitulant les informations sur sa réservation doit être donné à l'utilisateur, et une copie de ce billet doit être stockée sur la base de données.

SPÉCIFICATIONS

FONCTIONS

Fonctionnalités Métier

Fonctionnalités concernant le processus de recherche

Procédure principale `lance_recherche()` lançant les procédures et fonctions suivantes :

- Recherche de trajets contenant la gare désirée : `recherche_horaire()`
- Filtrage des trajets ne contenant pas les bonnes gares de départ et d'arrivée ou dans le mauvais sens : `compare_nodate()`
- Filtrage des trajets ne circulant pas le jour de la semaine découlant de la date de voyage désirée : `compare_avecdate()`
- Calcul du tarif des trajets filtrés : `trouve_tarif()`
- Quantification du nombre de places disponibles pour les trajets restants, en première et seconde classe : `verification_res_dispo()`
- Tri des trajets en fonctions de l'heure de départ : `tri()`
- Procédure principale de réservation : `reservation()`

Fonctionnalités concernant le processus de réservation

Procédure lançant les procédures et fonctions suivantes :

- Écriture des places réservées dans la structure contenant les places : `ecriture_resa_in_tab_places()`
- Procédure de paiement : `faux_paiement()`
- Écriture des billets créés dans la structure contenant le registre des billets : `creation_fichier_billet()`

Fonctionnalités Support

Les horaires

Récupération des horaires depuis un dossier vers une structure dédiée : `chargement_horaires()`

Les dates ouvertes à la vente

Création d'un tableau contenant les dates pour lesquelles la vente de billets est ouverte : `crea_datevente()`

Les Places

Fonctionnalités en Amont

- Récupération des places depuis une architecture de dossiers vers une structure dédiée : `chargement_places()`

- Tri de la structure contenant les places afin de supprimer celles concernant des trajets passés
- Incrémentation de la structure contenant les places afin d'ajouter celles concernant les dates et trajets désormais ouverts à la vente

Fonctionnalités en Aval

- Création et/ou écriture depuis la structure contenant les places vers une architecture de dossiers : `sauvegarde_places()`

Les Billets

- Récupération du registre des billets depuis un dossier vers une structure dédiée : `chargement_billets()`
- Écriture du registre des billets depuis une structure dédiée vers un dossier : `sauvegarde_billets()`

Fonctionnalités Outils

Fonctionnalités concernant les interprétations de code

- Interprétation de la position du siège, de son code vers son nom : `interprete_position()`
- Interprétation de la salle, de son code vers son nom : `interprete_salle()`

Fonctionnalités concernant les dates

- Saisie de date valide : `valide_date()`
- Récupération du jour de semaine depuis une date : `calcul_jour_semaine()`
- Interprétation du jour de semaine, de son code vers son nom : `interprete_jour_semaine()`
- Incrémentation/Décrémentation de la date : `date_suivante_precedente()`
- Antériorité d'une première date à une deuxième : `date_anterieure()`
- Séparation d'une date en jour, mois, année : `decoupe_date()`
- Assemblage d'une date au format entier AAAAMMJJ : `assemble_date()`

Fonctionnalités concernant les saisies

- Lecture de choix utilisateur sécurisée : `lecture_choix()`
- Saisie numérique sécurisée : `saisie_int()`
- Saisie textuelle sécurisée : `saisie_text()`
- Vidage du buffer : `dump_buffer()`

Fonctionnalités de traitement des chaines

- Conversion de chaque caractère d'une chaine de caractères en majuscule : `convmaj()`
- Conversion de chaque - en `space` d'une chaine de caractères : `tiret_to_space()`

- Coupe une chaine en deux chaines au caractère donné : `coupe_chaine_au_caractere()`
- Suppression d'une sous-chaine au début d'une chaine : `nettoie_debut_chaine()`
- Suppression des espaces aux début et fin de chaine : `trim()`
- Conversion de chaque motif dans une chaine de caractères en un autre motif : `str_replace_all()`
- Suppression des accents : `supprime_accent()`

Fonctionnalités concernant les interfaces avec le système

- Récupération du la date du système : `date_sys()`
- Récupération du contenu d'un répertoire : `ListerRep()`
- Récupération des informations sur un objet : `AfficheObjetInfo()`
- Suppression d'un répertoire : `supprime_repertoire()`

BASE DE DONNÉES

Jeux de données

Les ensembles de données utilisés sont ceux mis à disposition par la SNCF à https://ressources.data.sncf.com/explore/

- Horaires
  - Horaires des TGV : https://ressources.data.sncf.com/explore/dataset/horaires-des-train-voyages-tgvinouiouigo/table/
  - Horaires des lignes TER : https://ressources.data.sncf.com/explore/dataset/sncf-ter-gtfs/table/
  - Horaires des lignes Transilien : https://ressources.data.sncf.com/explore/dataset/sncf-transilien-gtfs/table/
  - Horaires des lignes Intercités : https://ressources.data.sncf.com/explore/dataset/sncf-intercites-gtfs/table/
  - Horaires des Tram-Train TER Pays de la Loire : https://ressources.data.sncf.com/explore/dataset/sncf-tram-train-ter-pdl-gtfs/table/
- Tarifs
  - Tarifs TGV par origine-destination : https://data.sncf.com/explore/dataset/tarifs-tgv-par-od/table/
  - Tarifs TER par origine-destination : https://data.sncf.com/explore/dataset/tarifs-ter-par-od/table/?disjunctive.region&sort=region
  - Tarifs Intercités 100% ECO : https://data.sncf.com/explore/dataset/tarifs-intercites-100-eco/table/?sort=origine
  - Tarifs Intercités de jour : https://data.sncf.com/explore/dataset/tarifs-intercites-de-jour/table/?sort=origine

Architecture des dossiers de données

Horaires

Les données relatives aux horaires sont situées dans le dossier `data`, puis le sous dossier `horaire`. Le chemin relatif est donc : `./data/horaire/`.

Tarifs

Les données relatives aux tarifs sont situées dans le dossier `data`, puis le sous dossier `tarif`. Le chemin relatif est donc : `./data/tarif/`.

Billets

Les données relatives aux billets sont situées dans le dossier `data`, puis le sous dossier `billet`. Le chemin relatif est donc : `./data/billet/`.

Places

Les données relatives aux places sont situées dans le dossier `data`, puis le sous dossier `place`, puis un des sous dossiers de trajets ayant comme nom l'identifiant du trajet, puis un des sous dossiers de date ayant comme nom une date au format AAAAMMJJ, puis un des sous dossiers de séquence ayant comme nom le numéro de la séquence. Le chemin relatif est donc : `./Data/Places/<idtrajet>/<date>/<sequence>/`.

Description des données

Horaires

Les ensembles de données sont disponibles au format GTFS (*General Transit Feed Specification*), format de fichier standardisé pour les horaires de transports en commun.

Description du format GTFS

Parmi les fichiers obligatoires et facultatifs du standard GTFS, chaque ensemble de données Horaires fourni par la SNCF est constitué de 8 fichiers :

- `agency.txt` : Agences de transports en commun ayant un service représenté dans cet ensemble de données.
- `stops.txt` : Arrêts où les voyageurs peuvent monter et descendre. Définit également les stations et leurs entrées.
- `routes.txt` : Itinéraires en transports en commun. Un itinéraire est un ensemble de trajets présentés aux voyageurs comme relevant du même service.
- `trips.txt` : Trajets pour chaque itinéraire. Un trajet est une série d'au moins deux arrêts desservis à des horaires précis.
- `stop_times.txt` : Heures d'arrivée et de départ d'un train depuis des arrêts spécifiques, pour chaque trajet.

- `calendar.txt` : Dates de service indiquées à l'aide d'un horaire hebdomadaire comportant des dates de départ et d'arrivée.
- `calendar_dates.txt` : Exceptions pour les services définis dans le fichier `calendar.txt`.
- `transfers.txt` : Règles de liaison aux pôles de correspondance entre des itinéraires.

Le schéma logique d'un ensemble de données Horaires est le suivant :

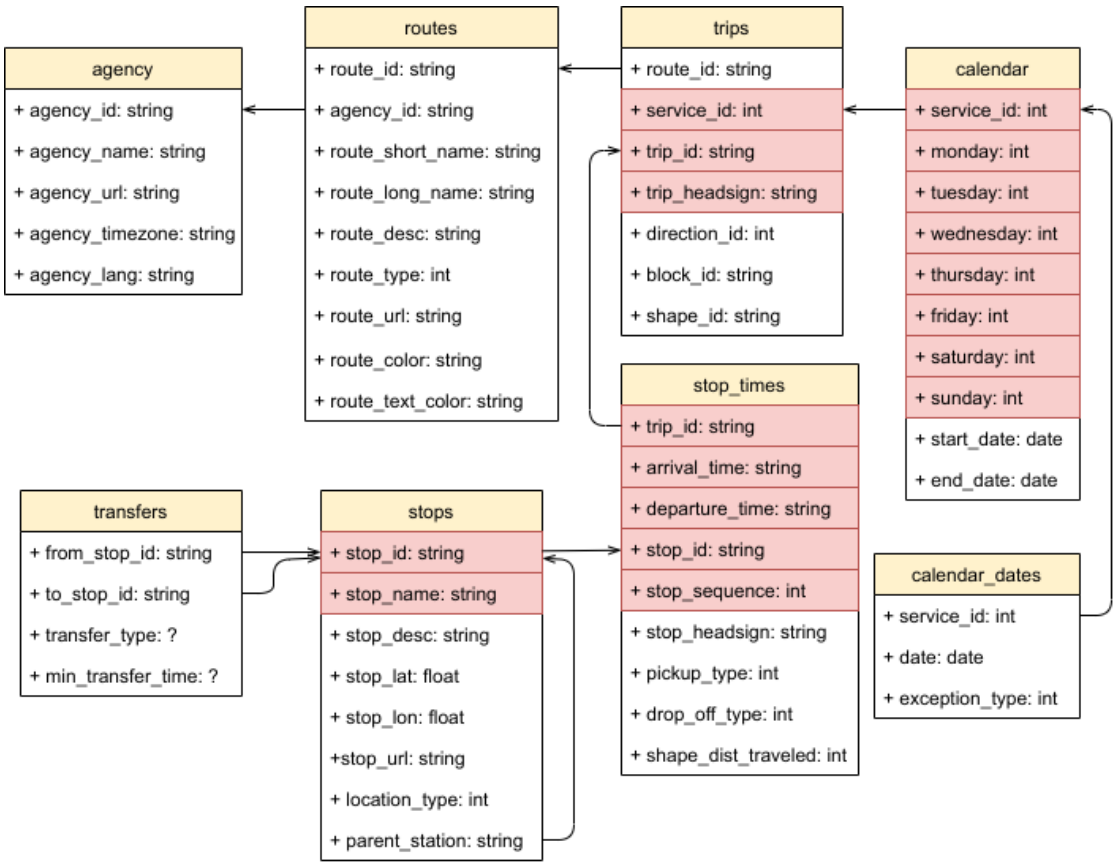

Schéma logique des ensembles de données Horaires fournis par la SNCF

Les champs de couleur rouge sont ceux traités par le programme.

Fichier `agency.txt`

Le fichier `agency.txt` décrit les agences de transports en commun ayant un service représenté dans cet ensemble de données. Il s'agit de la SNCF. Il comporte les champs :

- `agency_id` : indique la marque du réseau de transports en commun (souvent identique au nom de l'agence)
- `agency_name` : nom complet de l'agence de transports en commun.
- `agency_url` : URL de l'agence de transports en commun.
- `agency_timezone` : fuseau horaire de la zone où se trouve l'agence de transports en commun.

- `agency_lang` : langue principale utilisée par cette agence de transports en commun.

Fichier `stops.txt`

Le fichier `stops.txt` décrit les arrêts où les voyageurs peuvent monter et descendre, c'est à dire les gares. Il est lié à la table `stop_times.txt` par le champ `stop_id`. Il comporte les champs :

- `stop_id` : identifie un arrêt, une station ou une entrée de station. Le terme "entrée de station" désigne à la fois les entrées et les sorties de station. Les arrêts, les stations et les entrées de station sont collectivement appelés "emplacements". Le même arrêt peut être desservi par plusieurs itinéraires.
- `stop_name` : nom de l'emplacement.
- `stop_desc` : description de l'emplacement. Ce champ n'est pas toujours renseigné.
- `stop_lat` : latitude de l'emplacement.
- `stop_lon` : longitude de l'emplacement.
- `zone_id` : définit la zone tarifaire d'un arrêt. Ce champ n'est pas toujours renseigné.
- `stop_url` : URL d'une page Web qui décrit l'emplacement. Ce champ n'est pas toujours renseigné.
- `location_type` : type d'emplacement :
  - 0 ou vide : arrêt ou quai (lieu où les usagers montent dans un véhicule de transport en commun ou en descendent). Le terme "quai" est utilisé lorsque cette valeur est définie au sein d'un champ `parent_station`.
  - 1 : station (zone ou structure physique comprenant un ou plusieurs quais)
  - 2 : entrée ou sortie (lieu où les usagers peuvent entrer dans une station depuis la rue ou en sortir). Si une entrée/sortie appartient à plusieurs stations, tous les chemins correspondants sont indiqués, et le fournisseur de données doit désigner une station en tant que station principale (parente).
  - 3 : intersection générique (un emplacement dans une station qui ne correspond à aucune autre valeur `location_type`).
  - 4 : Zone d'embarquement (un emplacement spécifique sur un quai où les usagers peuvent monter à bord d'un véhicule ou en descendre)
- `parent_station` : spécifie la hiérarchie entre les différents emplacements définis dans le fichier `stops.txt`. Contient l'ID de l'emplacement parent, comme suit :
  - Arrêt/quai (`location_type=0`) : le champ `parent_station` contient l'ID d'une station.
  - Station (`location_type=1`) : ce champ doit être vide.
  - Entrée/sortie (`location_type=2`) ou intersection générique (`location_type=3`) : le champ `parent_station` contient l'ID d'une station (`location_type=1`).
  - zone d'embarquement (`location_type=4`) : le champ `parent_station` contient l'ID d'un quai.

Fichier routes.txt

Le fichier routes.txt décrit les itinéraires en transports en commun. Un itinéraire est un ensemble de trajets présentés aux voyageurs comme relevant du même service. Il est liée à agency.txt par le champ agency_id. Il comporte les champs :

- route_id : définit un itinéraire.
- agency_id : agence pour l'itinéraire spécifié. Fait référence à agency.agency_id.
- route_short_name : version courte du nom d'un itinéraire. Il s'agit généralement d'un identifiant court, abstrait, comme "32", "100X" ou "vert", que les usagers utilisent pour identifier un itinéraire, sans donner d'indications sur les lieux desservis
- route_long_name : le nom complet d'un itinéraire. Ce nom est généralement plus descriptif que la version courte indiquée dans le champ route_short_name. Il inclut souvent la destination ou le terminus de l'itinéraire.
- route_desc : description d'un itinéraire. Les informations fournies doivent être utiles et de qualité, elles ne répètent pas le nom de l'itinéraire. Ce champ n'est pas renseigné.
- route_type : décrit le moyen de transport utilisé pour un itinéraire. Les options suivantes sont acceptées :
  - 0 : tramway ou métro léger. Tout système de métro léger ou circulant sur la chaussée dans une zone métropolitaine.
  - 1 : métro. Tout système ferroviaire souterrain circulant au sein d'une zone métropolitaine.
  - 2 : train. Utilisé pour les trajets interurbains ou longue distance.
  - 3 : bus. Utilisé pour les lignes de bus courte et longue distance.
  - 4 : ferry. Utilisé pour le service de bateaux courte et longue distance.
  - 5 : tramway à traction par câble. Utilisé pour les systèmes de tramways au niveau de la chaussée dans lesquels le câble passe sous le véhicule, comme c'est le cas à San Francisco.
  - 6 : téléphérique. Service de transport par câble où les cabines, voitures, télécabines ou sièges sont suspendus à l'aide d'un ou de plusieurs câbles.
  - 7 : funiculaire. Tout système ferroviaire conçu pour les pentes raides.
  - 11 : trolleybus. Autobus électrique alimenté par des lignes aériennes de contact.
  - 12 : monorail. Service de chemin de fer roulant sur une voie constituée d'un rail ou d'une poutre unique.
- route_url : URL d'une page Web pour un itinéraire spécifique.
- route_color : couleur de l'itinéraire correspondant à celle utilisée dans les supports destinés au public.
- route_text_color : couleur lisible pour le texte à afficher sur la couleur d'arrière-plan route_color.

Fichier `trips.txt`

Le fichier `trips.txt` définit les trajets pour chaque itinéraire. Il est lié à `routes.txt` par le champ `route_id`. Il comporte les champs :

- `route_id` : définit un itinéraire. Fait référence à `routes.route_id`.
- `service_id` : définit les dates auxquelles le service est disponible pour un ou plusieurs itinéraires. Fait référence à `calendar.service_id` ou à `calendar_dates.service_id`
- `trip_id` : définit un trajet.
- `trip_headsign` : texte qui apparaît sur la signalétique indiquant aux voyageurs la destination du trajet (le numéro de train tel qu'affiché en gare et sur les billets). Ce champ permet de différencier les modèles de service sur un même itinéraire.
- `direction_id` : indique la direction du trajet. Ce champ n'est pas utilisé pour les itinéraires, mais il permet de distinguer les trajets en fonction de leur direction lors de la publication des horaires. Les options suivantes sont acceptées :
  - `0` : trajet dans un sens (trajet aller, par exemple).
  - `1` : trajet dans le sens opposé (trajet retour, par exemple)
- `block_id` : identifie le bloc auquel appartient le trajet. Un bloc comprend un seul trajet ou de nombreux trajets séquentiels effectués par le même véhicule. Ce champ n'est pas renseigné.
- `shape_id` : définit une forme géospatiale décrivant le parcours du véhicule lors d'un trajet. Ce champ n'est pas renseigné.

Fichier `stop_times.txt`

Le fichier `stop_times.txt` définit les heures d'arrivée et de départ d'un train depuis des arrêts spécifiques, pour chaque trajet. Il est lié à `stops.txt` par le champ `stop_id`. Il comporte les champs :

- `trip_id` : définit un trajet. Fait référence à `trips.trip_id`.
- `arrival_time` : heure d'arrivée à un arrêt donné pour un trajet spécifique inclus dans un itinéraire. Si les heures d'arrivée et de départ sont identiques pour un arrêt spécifique, la même valeur est saisie pour les champs `arrival_time` et `departure_time`. Pour les heures après minuit de la journée de service, la valeur saisie est une supérieure à 24:00:00 au format HH:MM:SS dans l'heure locale du jour où commence le trajet.
- `departure_time` : heure de départ depuis un arrêt donné pour un trajet spécifique inclus dans un itinéraire. Si les heures d'arrivée et de départ sont identiques pour un arrêt spécifique, la même valeur est saisie pour les champs `arrival_time` et `departure_time`. Pour les heures après minuit de la journée de service, la valeur saisie est supérieure à 24:00:00 au format HH:MM:SS de l'heure locale le jour où commence le trajet.
- `stop_id` : identifie l'arrêt desservi. Fait référence à `stops.stop_id`.
- `stop_sequence` : ordre des arrêts desservis lors d'un trajet particulier. Les valeurs augmentent à mesure du trajet, mais ne sont pas nécessairement consécutives.

- `stop_headsign` : texte qui apparaît sur la signalétique indiquant aux voyageurs la destination du trajet.
- `pickup_type` : indique les possibilités de montée à bord. Les options suivantes sont acceptées :
  - `0` ou vide : les usagers peuvent monter à bord aux horaires standards.
  - `1` : les usagers ne peuvent pas monter à bord.
  - `2` : les usagers doivent téléphoner à l'agence pour pouvoir monter à bord.
  - `3` : les usagers doivent contacter le conducteur pour pouvoir monter.
- `drop_off_type` : indique les possibilités de descente du véhicule. Les options suivantes sont acceptées :
  - `0` ou vide : les usagers peuvent monter à bord aux horaires standards.
  - `1` : les usagers ne peuvent pas monter à bord.
  - `2` : les usagers doivent téléphoner à l'agence pour pouvoir monter à bord.
  - `3` : les usagers doivent contacter le conducteur pour pouvoir monter.
- `shape_dist_traveled` : indique la distance réelle parcourue le long du tracé donné entre le premier arrêt et l'arrêt spécifié dans cet enregistrement.

Fichier `calendar.txt`

Le fichier `calendar.txt` définit les dates de service indiquées à l'aide d'un horaire hebdomadaire comportant des dates de départ et d'arrivée. Il est lié au fichier `trips.txt` par le champ `service_id`. Il comporte les champs :

- `service_id` : définit de façon unique les dates auxquelles le service est disponible pour un ou plusieurs itinéraires. Fait référence à `trips.service_id`.
- `monday` : indique si le service est proposé tous les lundis de la plage de dates spécifiée par les champs `start_date` et `end_date`. Les options suivantes sont acceptées :
  - `1` : le service est disponible tous les lundis de la plage de dates.
  - `0` : le service n'est pas disponible les lundis de la plage de dates.
- `tuesday` : fonctionne comme le champ `monday`, mais pour les mardis
- `wednesday` : fonctionne comme le champ `monday`, mais pour les mercredis
- `thursday` : fonctionne comme le champ `monday`, mais pour les jeudis
- `friday` : fonctionne comme le champ `monday`, mais pour les vendredis
- `saturday` : fonctionne comme le champ `monday`, mais pour les samedis
- `sunday` : fonctionne comme le champ `monday`, mais pour les dimanches
- `start_date` : date de début de validité du service
- `end_date` : date de fin de validité du service

Fichier `calendar_dates.txt`

Le fichier `calendar_dates.txt` définit les exceptions pour les services définis dans le fichier `calendar.txt`. Il est lié au fichier `calendar.txt` par le champ `service_id`. Il comporte les champs :

- `service_id` : définit les dates auxquelles le service est exceptionnellement disponible ou indisponible pour un ou plusieurs itinéraires. Fait référence à `calendar.service_id`.
- `date` : date à laquelle le service proposé est différent du service standard.
- `exception_type` : indique si le service est disponible à la date spécifiée dans le champ date. Les options suivantes sont acceptées :
  - `1` : le service a été ajouté pour la date spécifiée.
  - `2` : le service a été supprimé pour la date spécifiée.

Nous ne nous servons pas de ce fichier.

Fichier `transferts.txt`

Le fichier `transferts.txt` spécifie des règles et des valeurs de remplacement supplémentaires pour les correspondances sélectionnées. Il comporte les champs :

- `from_stop_id` : identifie l'arrêt ou la station de départ pour une liaison entre deux itinéraires. Si ce champ fait référence à une station, la règle de correspondance s'applique à tous ses arrêts enfants. Fait référence à `stops.stop_id`.
- `to_stop_id` : identifie l'arrêt ou la station de départ pour une liaison entre deux itinéraires. Si ce champ fait référence à une station, la règle de correspondance s'applique à tous ses arrêts enfants. Fait référence à `stops.stop_id`.
- `transfer_type` : Indique le type de correspondance pour la paire (from_stop_id, to_stop_id) spécifiée. Les options suivantes sont acceptées :
  - `0` ou vide : point de correspondance recommandé entre deux itinéraires.
  - `1` : point de correspondance temporisé entre deux itinéraires. Le véhicule qui part doit attendre celui qui arrive et laisser suffisamment de temps pour que les usagers puissent prendre la correspondance.
  - `2` : correspondance nécessitant une durée minimale entre l'heure d'arrivée et l'heure de départ. Spécifiez la durée en question dans le champ `min_transfer_time`.
  - `3` : aucune correspondance ne peut être assurée à cet emplacement.
- `min_transfer_time` : délai (en secondes) devant être accordé pour permettre une correspondance entre deux itinéraires aux arrêts spécifiés.

Le fichier est vide pour les ensembles de données utilisés. Nous ne nous servons pas de ce fichier.

Données traitées par le programme

Toutes les données fournies dans un ensemble de données Horaires ne sont pas utiles pour le programme `reservation`. Parmi les données des ensembles de données, seuls certains fichiers sont lus par le programme : `trips.txt`, `calendar.txt`, `stop_times.txt` et `stops.txt`. Parmi les champs des fichiers lus par le programme, seuls certains sont stockés en mémoire : les champs de couleur rouge.

Étant donné que notre programme ne rencontrera pas de modifications des fichiers concernant la circulation, nous avons choisi de na pas gérer les dates de début et de fin de validité d'un calendrier. Chaque trajet opère donc pour une durée illimitée.

Un trajet dans le programme peut donc être décrit comme suit :

> Un trajet effectuant la liaison depuis Bordeaux vers Nice en passant par Marseille, dans cet ordre, partant de Bordeaux à 14h03 et arrivant à Nice à 18h47, opérant tous les mardis et tous les mercredis.

Par ailleurs, les données ont été échantillonnées pour réduire le volume à traiter. Seul l'ensemble de données horaires TGV a été utilisé. Il a également été réduit.

Places

Les fichiers `places.txt` sont tous identiques. Ils contiennent 6 colonnes :

- `wagon` : le numéro du wagon dans lequel se trouve la place : allant de 1 à 8.
- `classe` : la classe à laquelle appartient le wagon : allant de 1 à 2.
- `salle` : la salle à laquelle se trouve la place : allant de 0 à 1.
- `siege` : le numéro de siège de la place.
- `position` : la position de la place : allant de 0 à 2.
- `disponible` : prend la valeur du billet qui a réservé ce siège, ou 0 si disponible.

Ils sont construits du fichier `./data/train/placement_tgv_duplex_echantillon.txt` qui est un échantillonnage du fichier `./data/train/placement_tgv_duplex.txt`(TGV duplex génération 1).

Structure Places

La structure contenant les places est articulée ainsi :

- `idtrajet` : l'identifiant du trajet
- `type` : le type du train
- `date[]` -> un tableau des dates pour lesquelles ce trajet circule
  - `date` : la date au format AAAAMMJJ
  - `sequence[]` -> un tableau des sequences du trajet
    - `gd`: la gare de départ de la séquence
    - `ga` : la gare d'arrivée de la séquence
    - `seqdep` : la position de la gare de départ dans son trajet
    - `seqarr` : la position de la gare d'arrivée dans son trajet
    - `hd` : l'heure de départ de la séquence
    - `ha` : l'heure d'arrivée de la séquence
    - `place[]` -> un tableau des places (disponibilité et caractéristiques) pour cette séquence
      - `wagon`

- classe
- salle
- siege
- position
- billet

Tarifs

La structure contenant les tarifs est articulée ainsi :

- `garedep` : la gare de départ ou d'arrivée
- `garearr`: la gare d'arrivée ou de départ
- `prix0` : le tarif d'appel en seconde classe
- `prix2`: le plein tarif en seconde classe
- `prix1`: le plein tarif en première classe

Elle est construite à partir du fichier `./data/tarif/tarifs-tgv-par-od.csv`.

Billets

Le fichier `registre_billets.txt` contient les informations suivantes réparties dans X colonnes :

- `idtrajet` : une chaine de caractères identifiant le trajet.
- `garedep` : une chaine de caractères identifiant la gare de départ par son nom.
- `garearr` : une chaine de caractères identifiant la gare d'arrivée par son nom.
- `nom` : une chaine de caractères contenant le nom du passager.
- `prenom` : une chaine de caractères contenant le prenom du passager.
- `date` : la date de voyage au format AAAAMMJJ.
- `age` : l'age du passager.
- `classe` : la classe du wagon dans lequel le passager voyage.
- `wagon` : le numéro du wagon dans lequel le passager voyage.
- `salle` : la salle du wagon dans lequel le passager voyage.
- `siege` : le numéro du siège dans lequel le passager voyage.
- `position` : la position du siège dans lequel le passager voyage.
- `prix` : le prix que le passager a payé pour voyager.
- `billet` : le code du billet.

PROCESSUS

Le programme démarre par la fonction `main`, qui appelle automatiquement des procédures, puis affiche un menu à l'utilisateur, lui proposant un choix d'options.

Processus Support Amont

Deux procédures sont appelées par `main` avant d'afficher un menu à l'utilisateur :

- `date_sys()` : récupération de la date système
- `chargement()` : chargement des données à traiter

Date système

La date système est une information utilisée par le programme. Elle est récupérée par la procédure `date_sys()` dans les variables `jour_sys`, `mois_sys`, `annee_sys` et `j_hebdo_num_sys`.

Chargement des données

La procédure `chargement()` appelle une série de procédures de chargement et de calcul des données utilisées par le programme :

- `chargement_horaires()` : chargement des données horaires des trains
- `chargement_tarifs()` : chargement des données tarifaires des voyages en train
- `crea_date_vente()` : création d'un tableau de dates dont les places de train sont ouvertes à la vente
- `chargement_places()` : chargement des données relatives aux places des trains ouverts à la vente
- `chargement_billets()` : chargement des données relatives aux places réservées.

Chargement des données horaires

La procédure `chargement_horaire()` appelle 4 procédures de chargement de données correspondant aux 4 fichiers traités des ensembles de données horaires :

- `chargement_trajet()` : chargement des données de trajet des trains
- `chargement_calendrier()` : chargement des données de calendrier de service des trajets des trains
- `chargement_stop()` :chargement des données horaires d'arrêt des trains en gare
- `chargement_gare()` : chargement des données des gares du réseau.

Chargement des données de trajet

La procédure `chargement_trajet()` ouvre le fichier `./data/horaire/trips.txt` et charge dans le tableau `trajets[]` les données suivantes :

- `idtrajet` : l'identifiant du trajet
- `idservice` : l'identifiant de son calendrier de service
- `numtrain` : le numéro de train de la signalétique
- `direction` : le sens du trajet

Chargement des données de calendrier de service

La procédure `chargement_calendrier()` ouvre le fichier `./data/horaire/calendar.txt` et charge dans le tableau `calendrier[]` les données suivantes :

- `idservice` : l'identifiant du calendrier de service
- `lun, mar, mer, jeu, ven, sam, dim` : la valeur booléenne de circulation des jours hebdomadaires.

Chargement des données horaires d'arrêt des trains en gare

La procédure `chargement_stop()` ouvre le fichier `./data/horaire/stop_times.txt` et charge dans le `tableau horaires[]` les données suivantes :

- `sequence`
- `idgare` : l'identifiant de la gare d'arrêt
- `hd` : heure de départ
- `ha` : heure d'arrivée

Chargement des données de gares

La procédure `chargement_gare()` ouvre le fichier `./data/horaire/stops.txt` et charge dans le tableau `gares[]` les données suivantes :

- `idgare` : l'identifiant de la gare
- `nomgare` : le nom de la gare.

Chargement des données tarifaires

La procédure `chargement_tarifs()` ouvre le fichier `./data/tarif/tarifs_tgv_par_od.csv` et charge dans le tableau `tarifs[]` les données suivantes :

- `garedep` : le nom de la gare d'origine
- `garearr` : le nom de la gare de destination
- `prix0` : le tarif d'appel en seconde classe
- `prix2` : le plein tarif en seconde classe
- `prix0` : le tarif en première classe.

Création du tableau des dates ouvertes à la vente

La procédure `crea_date_vente()` crée le tableau `tab_date_vente[]` des dates de circulation des trains ouverts à la réservation.
Elle utilise un paramètre donné en nombre de mois (1 mois ici), calcule la date limite d'ouverture des ventes et incrémente les éléments de date depuis la date système, jusqu'à la date limite.
Elle convertit et stocke chaque date en une donnée de type `integer` au format AAAAMMJJ.

Chargement des données Places

La procédure `chargement_places()` lit les données du dossier `./data/place/` et les charge dans le tableau `tab_places[]`.

Elle liste les dossiers présents dans `./data/place/` :

- S'il n'y en a aucun, elle crée tous les dossiers `./data/place/<idtrajet>` où `<idtrajet>` est un item de `trajet[].idtrajet`
- S'il en existe, elle compare les dossiers existants et `trajet[].idtrajet`
- S'il manque un dossier `<idtrajet>`, elle le crée.
- Elle liste les dossiers présents.

Pour chaque dossier `./data/place/<idtrajet>`, la procédure lit et mémorise le fichier `./data/train/placement_[…].txt` du type de train correspondant.

La procédure liste les dossiers présents dans `./data/place/<idtrajet>`.

- Elle vérifie, pour chaque date de `tab_date_vente[]`, si le train de l' `<idtrajet>` circule à cette date :
  - Si le train circule
    - Si `./data/place/<idtrajet>` contient des dossiers
      - Si `./data/place/<idtrajet>/<date>` n'existe pas, elle le crée
    - Si `./data/place/<idtrajet>` ne contient pas de dossiers, elle crée `/data/place/<idtrajet>/<date>`
- Elle liste les dossiers présents dans `./data/place/<idtrajet>`

Pour chaque dossier `./data/place/<idtrajet>/<date>`, la procédure compare la date et `tab_date_vente[0].date`.

- Si `<date>` est inférieur à `tab_date_vente[0].date`, elle supprime `./data/place/<idtrajet>/<date>`.

La procédure liste les dossiers de `./data/place/<idtrajet>/<date>`.

- S'il est vide, pour chaque séquence de `sequence[<idtrajet>].garedep`, elle crée un fichier `./data/place/<idtrajet>/<date>/<séquence>/places.txt` où places.txt prend les valeurs de `placement_[…].txt`.
- Elle liste les fichiers de `./data/place/<idtrajet>/<date>/<sequence>`

La procédure lit le fichier `./data/place/<idtrajet>/<date>/<sequence>/places.txt` et charge dans le tableau des places `tab_places.[<idtrajet>].date[<date>].sequence[<sequence>].place[]` les données suivantes :

- `idtrajet` : l'identifiant du trajet
- `type` : le type du train
- `date` : la date du trajet
- `gd` : la gare de départ de la séquence
- ga : la gare d'arrivée de la séquence
- `seqdep` : le numéro de séquence à la gare de départ
- `seqarr` : le numéro de séquence à la gare d'arrivée
- `hd` : l'heure de départ de la séquence
- `ha` : l'heure d'arrivée de la séquence
- `wagon` : le numéro de wagon de la place
- `classe` : la classe du wagon de la place
- `salle` : la salle de la place
- `siege` : le numéro de siège de la place

- `position` : le placement de la place
- `billet` : le numéro de billet de la place

Chargement des billets

La procédure chargement_billets() ouvre le fichier `./data/registre_billet.txt` et charge dans le `tableau registre_billets[]` les données suivantes :

- `idtrajet` : l'identifiant du trajet
- `gd` : la gare de départ du voyage
- ga : la gare d'arrivée du voyage
- `nom` : le nom du voyageur
- `prenom` : le prénom du voyageur
- `date` : la date du trajet
- age : l'âge du voyageur
- `wagon` : le numéro de wagon de la place
- `classe` : la classe du wagon de la place
- `salle` : la salle de la place
- `siege` : le numéro de siège de la place
- `position` : le placement de la place
- `billet` : le numéro de billet de la place

Processus Métier

Après que les données ont été importées, le menu offre deux choix :

- Réserver qui appelle `lance_recherche()`
- Quitter qui appelle `quitter()`

Recherche

La procédure `lance_recherche()` appelle les procédures et fonctions métier, les invites de saisie utilisateur et l'affichage des résultats.

Saisies utilisateur et recherches itératives

La procédure `lance_recherche()`demande à l'utilisateur de saisir la gare de départ via une saisie sécurisée.
La procédure lance la fonctionnalité de recherche d'horaire de passage d'un trajet dans la gare de départ (fonction `lance_recherche()`). S'il n'y a pas de résultat, la procédure de recherche se termine ici.
La procédure demande à l'utilisateur de saisir la gare d'arrivée via une saisie sécurisée.
La procédure lance à nouveau la fonctionnalité de recherche d'horaire de passage d'un trajet dans la gare d'arrivée (fonction `lance_recherche()`). S'il n'y a pas de résultat, la procédure de recherche se termine ici.

La procédure lance la fonctionnalité de comparaison des trajets, ne conservant que les trajets reliant les gares de départ et d'arrivée, dans le bon sens de circulation (fonction `compare_nodate()`). S'il n'y a pas de résultat, la procédure de recherche se termine ici.

La procédure demande à l'utilisateur de saisir la date de voyage via une saisie sécurisée.

La procédure appelle des procédures de traitement de la date (procédures `interprete_jour_semaine()` et `assemble_date()`).

La procédure lance la fonctionnalité de filtrage des trajets en fonction du jour de semaine de la date de voyage renseignée (fonction `compare_avecdate()`). S'il n'y a pas de résultat, la procédure de recherche se termine ici.

La procédure lance la fonctionnalité de quantification des places disponibles pour chaque trajet, pour la première et la seconde classe (procédure `verification_res_dispo()`).

La procédure lance la fonctionnalité de tri des résultats par heure de départ des trajets (procédure `tri()`).

Affichage des résultats et choix utilisateur

La procédure procède à l'affichage des résultats sous forme de tableau.

La procédure affiche un menu et demande à l'utilisateur de choisir entre :

- Réserver un des trajets présentés
  - La procédure demande à l'utilisateur de choisir un des trajets proposés
  - La procédure lance la procédure de réservation pour le trajet choisi (procédure `reservation()`).
- Afficher les résultats pour le jour précédent
  Cet affichage n'est disponible que si la date est supérieure à la date système.
  - La procédure lance la fonctionnalité de décrémentation de la date (procédure `date_suivante_precedente()`)
  - La procédure traite la nouvelle date (procédures `interprete_jour_semaine()` et `assemble_date()`)
  - La procédure lance la fonctionnalité de filtrage des trajets en fonction du jour de semaine de la nouvelle date de voyage (fonction `compare_avecdate()`).
    - S'il n'y a pas de résultat, la procédure de recherche se termine ici
  - La procédure lance la fonctionnalité de quantification des places disponibles pour chaque trajet, pour la première et la seconde classe (procédure `verification_res_dispo()`).
  - La procédure lance la fonctionnalité de tri des résultats par heure de départ des trajets (procédure `tri()`).
  - La procédure boucle sur l'affichage des résultats sous formes de tableau, affiche le menu sous le tableau des résultats et demande à l'utilisateur de choisir de nouveau
- Afficher les résultats pour le jour suivant
  Cet affichage n'est disponible que si la date est inférieure à la dernière date des dates de trajets ouverts à la réservation
  - La procédure lance la fonctionnalité d'incrémentation de la date (procédure `date_suivante_precedente()`)

    - La procédure traite la nouvelle date (procédures `interprete_jour_semaine()` et `assemble_date()`)
    - La procédure lance la fonctionnalité de filtrage des trajets en fonction du jour de semaine de la nouvelle date de voyage (fonction `compare_avecdate()`).
        - S'il n'y a pas de résultat, la procédure de recherche se termine ici
    - La procédure lance la fonctionnalité de quantification des places disponibles pour chaque trajet, pour la première et la seconde classe (procédure `verification_res_dispo()`).
    - La procédure lance la fonctionnalité de tri des résultats par heure de départ des trajets (procédure `tri()`).
    - La procédure boucle sur l'affichage des résultats sous formes de tableau, affiche le menu sous le tableau des résultats et demande à l'utilisateur de choisir de nouveau
- Quitter la procédure de recherche

Réservation

La procédure de réservation `reservation()` demande des saisies utilisateur, procède à des vérifications relatives à la disponibilité des places par rapport aux choix utilisateur, et lance les procédures d'écriture dans la structure des places et dans la structure des billets.

Saisies utilisateur et sélection de place(s)

La procédure demande à l'utilisateur dans quelle classe il souhaite voyager via une saisie sécurisée.

- L'utilisateur a la possibilité de revenir au menu des résultats

La procédure demande à l'utilisateur de choisir le nombre de sièges qu'il souhaite réserver, dans la limite du nombre de places disponibles pour la classe qu'il a choisie.

- L'utilisateur a la possibilité de revenir au menu des résultats

La procédure recherche la disponibilité de la place pour chaque siège demandé.

La procédure demande à l'utilisateur de saisir les informations de chaque voyageur : nom, prénom, âge.

Si l'utilisateur a choisi la première classe :

- La procédure affiche un tableau des places disponibles en première classe et demande à l'utilisateur de choisir une des places proposées
    - L'utilisateur a la possibilité de revenir au menu des résultats

Si l'utilisateur a choisi la seconde classe :

- La procédure propose à l'utilisateur de choisir entre une salle (salle haute, salle basse) si des places sont disponibles dans les deux. Sinon elle affiche qu'il ne reste que des places dans une des salles.
- La procédure propose à l'utilisateur de choisir une position de siège (fenêtre, couloir) si des places sont disponibles pour les deux positions. Sinon elle affiche qu'il ne reste que des places pour une des positions.
- La procédure demande à l'utilisateur si ses choix, ou ces assignations lui conviennent, ou s'il préfère revenir au menu des résultats

La procédure affiche le montant total de la réservation

- L'utilisateur a la possibilité de revenir au menu des résultats

La procédure lance la fonctionnalité de paiement (procédure `faux_paiement()`).
Une fois le paiement effectué avec succès, la procédure lance la fonctionnalité d'écriture des réservations dans la structure contenant les places (procédure `ecriture_resa_in_tab_places()`).
La procédure lance ensuite la fonctionnalité d'écriture des billets dans la structure de registre des billets (procédure `creation_fichier_billets()`).

Processus Support Aval

Quitter

La procédure `quitter()` appelle la procédure de sauvegarde `sauvegarde()`, puis affiche un message d'au revoir.

Sauvegarde

La procédure `sauvegarde()` appelle les différentes procédures de sauvegarde :

- `sauvegarde_places()` : sauvegarde des données relatives aux places des trains ouverts à la vente
- `sauvegarde_billets()` : sauvegarde des données relatives aux places réservées.

Sauvegarde des places

La procédure `sauvegarde_places()` écrit les données du tableau `tab_places[]` dans les fichiers `./data/place/<idtrajet>/<date>/<sequence>/places.txt`.

Sauvegarde des billets

La procédure `sauvegarde_billets()()` écrit les données du tableau `registre_billets[]` dans le fichier `./data/registre_billet.txt`.

LIMITES

Plusieurs limites ont été rencontrées lors du développement. Elles résultent principalement des données d'entrée du programme.

Les données horaires sont peu volumineuses dans leur format de stockage GTFS. Cependant, une place est un siège, libre ou occupé, pour une séquence d'un trajet à une date donnée sur une période d'au moins 4 mois (cas des TGV). Le nombre de places à gérer est trop important pour que nos terminaux traitent l'entièreté des données horaires SNCF, et même l'entièreté d'un seul ensemble de données horaires.

Pour contourner ce problème, les choix suivants ont été faits :

- utiliser un seul ensemble de données horaires (TGV)
- réduire cet ensemble
- réduire la taille d'un train (45 places)
- réduire la période d'ouverture des ventes (1 mois)

Certaines incohérences dans le programme résultent d'un mauvais échantillonnage des données horaires (des trajets qui circulent à une date mais pour lequel il n'y a pas de séquences) : il existe des dossiers de dates sans séquences.

***

Un choix de simplification a été fait concernant les dates de début et fin de validité des services (elles sont ignorées). De ce choix résulte la présence de doublons dans les résultats de recherche : ils correspondent à des intervalles de validité du service différents.

***

La recherche d'un voyage au départ ou à l'arrivée d'une gare se fait par recherche de la sous-chaine « nom de gare » saisie par l'utilisateur dans la chaine « nom de gare » provenant des données horaires. Il arrive que la sous-chaine soit contenue dans un nom de gare ne désignant pas la gare recherchée. Par exemple, des résultats au départ ou à l'arrivée de la gare « Paris Gare de Lyon » apparaissent lorsque l'utilisateur souhaite voyager par Lyon.

Un choix de gare parmi une liste de gares trouvées sur la base de la saisie aurait pu être implémenté. Cette solution engendrait d'autres difficultés : la possibilité laissée à l'utilisateur de ne choisir qu'une seule gare. Comme exposé plus haut, certaines villes possèdent plusieurs gares que l'utilisateur veut voir apparaitre comme résultat d'une même recherche. Une solution consistait alors à faire choisir à l'utilisateur, pour chaque gare trouvée, s'il veut ou non l'inclure dans la recherche.

***

Un défaut du programme est lié à la qualité des données de tarif. Les noms de gare des données tarifaires sont généralement réduits au nom de la ville, tandis que les résultats de recherche contiennent le nom entier de la gare extrait des données horaires. Le choix a donc été fait de prendre les noms de gare des données tarifaires comme sous-chaîne et les noms de gare des données horaires comme chaîne lors de la recherche des tarifs de chaque résultat. Cependant, certains noms de gare restent différents comme ceux composés de « Saint », parfois orthographié « ST », Dans ces cas-là, la recherche échoue.

***

Un défaut de développement résulte du choix d'utiliser les données horaires des TGV, en oubliant de traiter le type de train dans plusieurs fonctions et procédures. Un élargissement de la portée du programme pour intégrer les données d'autres types de train nécessiterait des modifications.