

S3.01 - Développement d’application et Gestion de Projet

Enseignant tuteur : Nathalie VALLES-PARLANGEAU

Groupe n°1

**SAE du Troisième Semestre**

- Pôle Développement -

Spécifications Externes de notre problème algorithmique



Sport Track

Application Web Responsive de gestion de clubs sportifs amateurs

Matis Chabanat | Titouan Cocheril

Arthur Le Menn | Ivan Salle

- - -

TD1 | TP1

BUT Informatique

Semestre 3

- 2022 | 2023 -

Rappel du pitch de notre application

Dans le cadre de ce projet, nous avons décidé d’étudier le thème du sport et plus particulièrement de la gestion du sport amateur.

On retrouve sur internet beaucoup d’applications concernant le sport professionnel qui proposent différentes fonctionnalités telles que la visualisation des résultats ou de la composition des équipes, etc. Mais on en trouve très peu concernant le sport amateur. C’est dans le but de rendre la gestion des clubs sportifs amateur plus simple que nous avons décidé de créer “Sport Track” l’application qui centralise toute la vie du club au même endroit.

Nous souhaitons proposer un produit qui permettra de faire vivre le sport amateur en le rendant accessible à tout sportif licencié dans un club Français.

L’application prendra la forme d’un site web-responsive, c’est-à-dire que l’on pourra y accéder sur n’importe quel navigateur web et donc sur n’importe quel appareil (smartphone, ordinateur). Le produit sera également gratuit car le but est de rendre le sport amateur accessible et visible à tous.

Enfin, nous avons misé sur un style simple et épuré ce qui permettra d’éviter de perdre l’utilisateur avec des interfaces trop complexes.

Données manipulées



**Description des tables :**

Joueurs : Tous les joueurs licenciés

Equipes : Toutes les équipes déclarées

Championnats : Tous les championnats

Matchs : Tous les matchs

Stade : Les stades dans lesquels ont lieu les matchs et entraînements.

evenementsMatch : Les évènements qui ont lieux durant le match (but, carton etc)

jouerMatch : lien des joueurs et des matchs

participerChampionnat : lien entre équipe et championnat.

Messages : Les messages

Entrainement - Autre évènements : Tous les évènements internes des clubs

Inscrit : Tous les joueurs licenciés inscrits

**Problème Algorithmique : Générateur de QR Code**

Pitch du problème

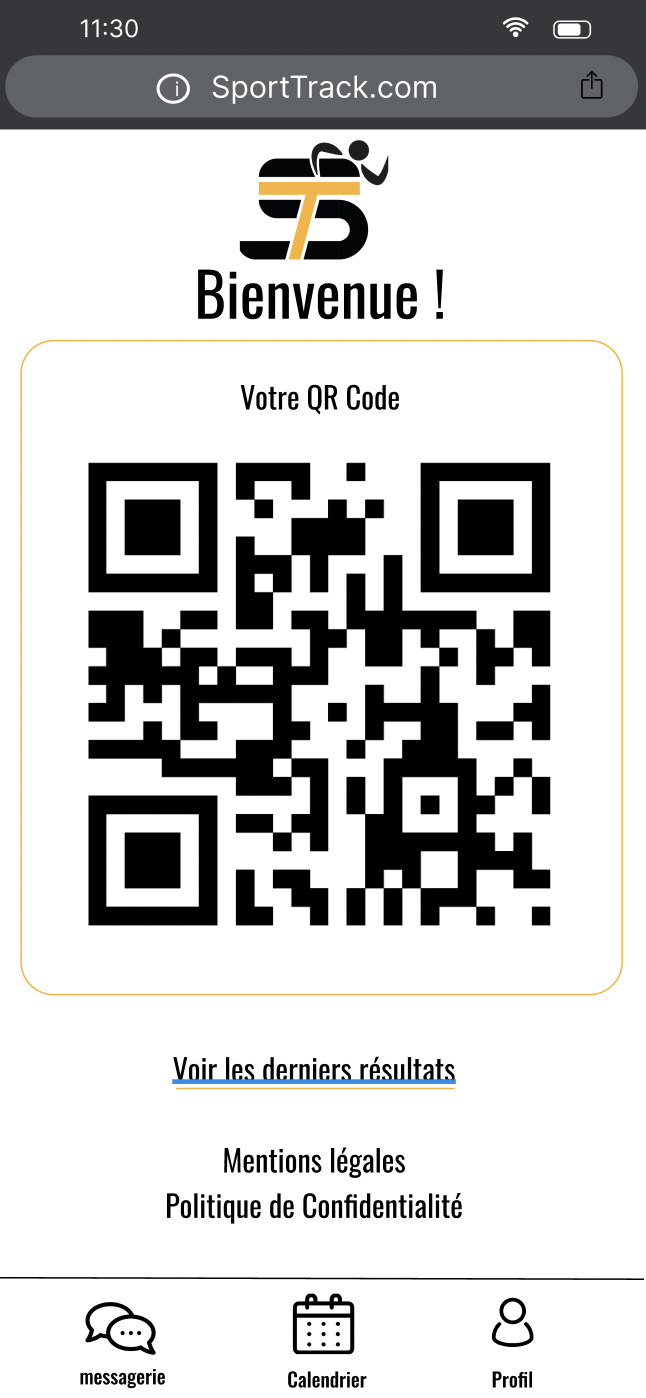
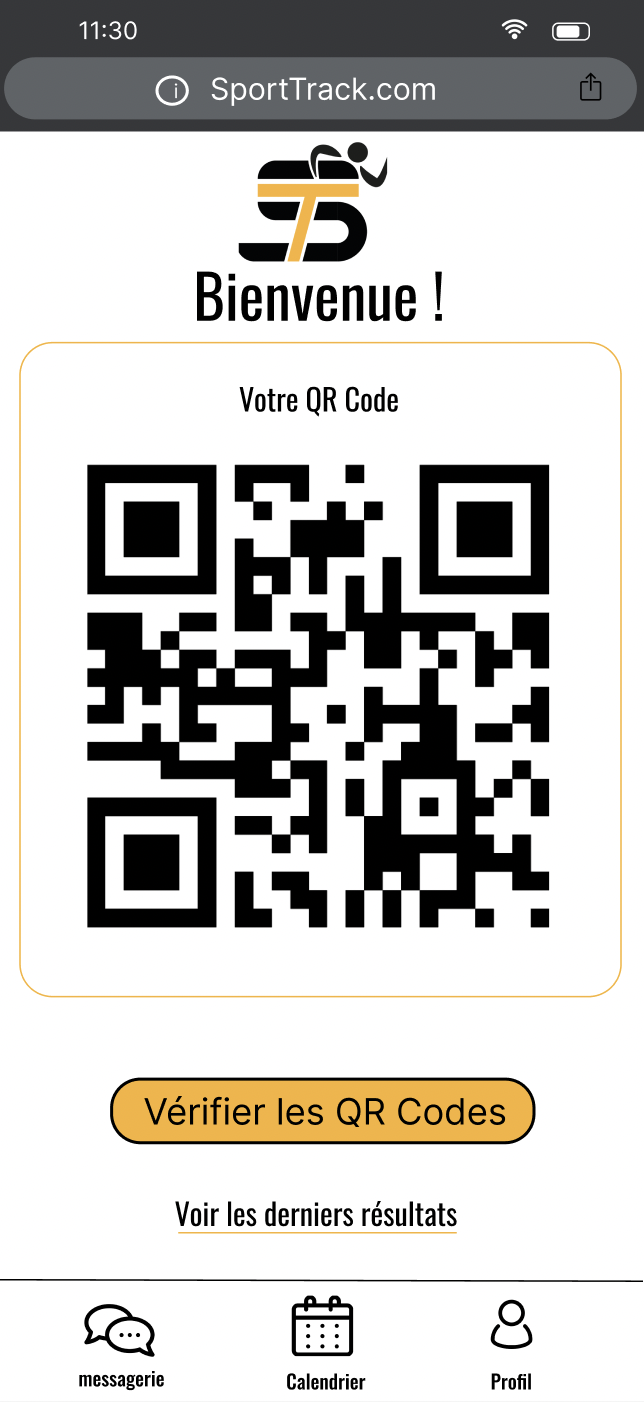
Sur la page d’accueil le joueur trouvera un QR code qu’il pourra venir présenter à l'entraînement et au match. Pour signifier sa présence et pour prouver que c’est bien le joueur possédant la licence et donc la vraie personne.

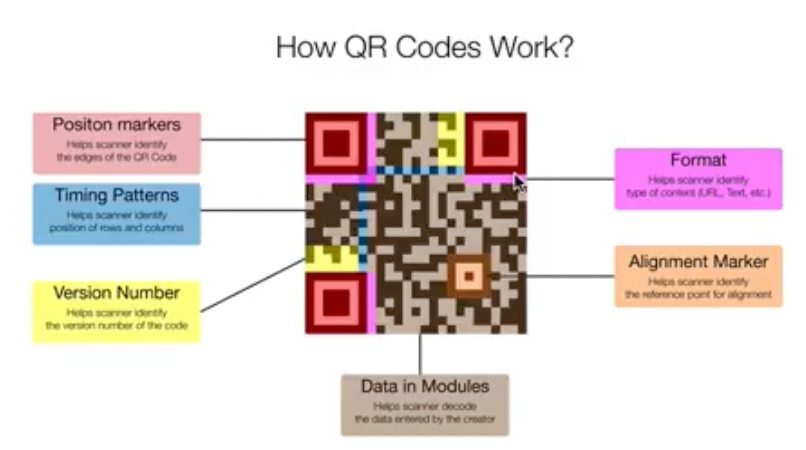
L'entraîneur trouvera sur sa page d’accueil un bouton de vérification de présence de ses joueurs qui accède à l’appareil photo et peut scanner. Il trouvera également son QR code pour signifier sa présence aux matchs.

L’idée Algorithmique : à partir des licences (données) joueurs, entraîneurs générer des QR code (résultats) et pouvoir les vérifier via l’appareil photo (une vérification entre le message retourné par le QR code et une base de données de toutes les licences sera effectuée).



Maquettes / schémas illustrant le problème



Voir annexe pour comprendre comment fonctionne un QR code

Informations manipulées pour traiter le problème

Il existe deux types de QR code statique et dynamique en fonction de si on veut ou non modifier les données entrées. Nos données entrées seront les licences de joueurs et entraîneur, nôtre QR code sera donc statique.

Le résultat sera un QR code généré par la licence.

Une fois scanné le QR code modifie la base de données des présences dans l’équipe, affiche sur l’écran du scanner le succès du scan

Suite de réflexion :

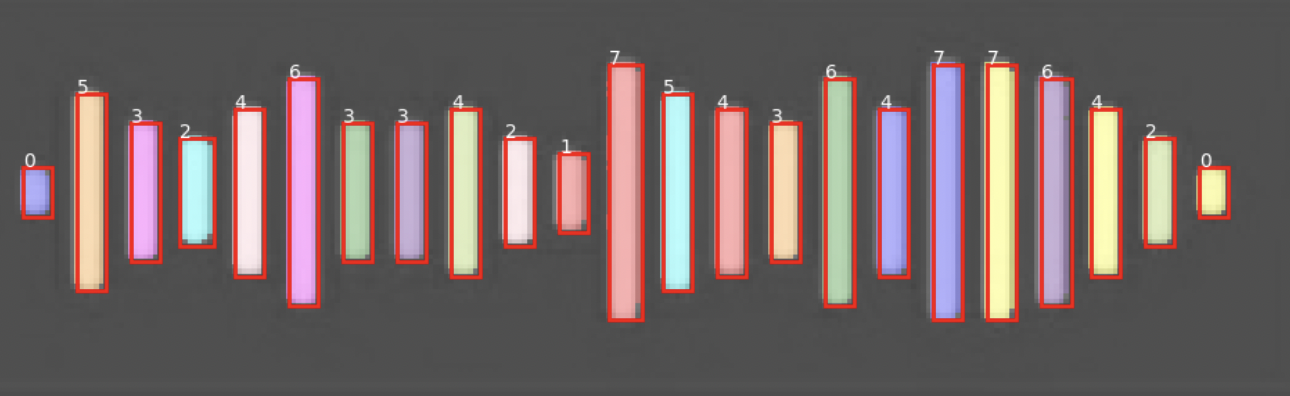
Le QR code était notre première idée pour résoudre le problème algorithmique mais il a néanmoins des avantages/défauts comme le fait qu’il puisse être scanné par tous les appareils, c’est pratique car pas besoin de développer un scanner mais ça pose un problème de sécurité à notre application.

En effet , n'importe qui de non inscrit/connecté à l’application peut obtenir les informations détenu dans le QR code.

On a donc choisi d’encoder nos informations dans un code barres (type spotify) et de développer notre propre scanner.

Comment ça marche :

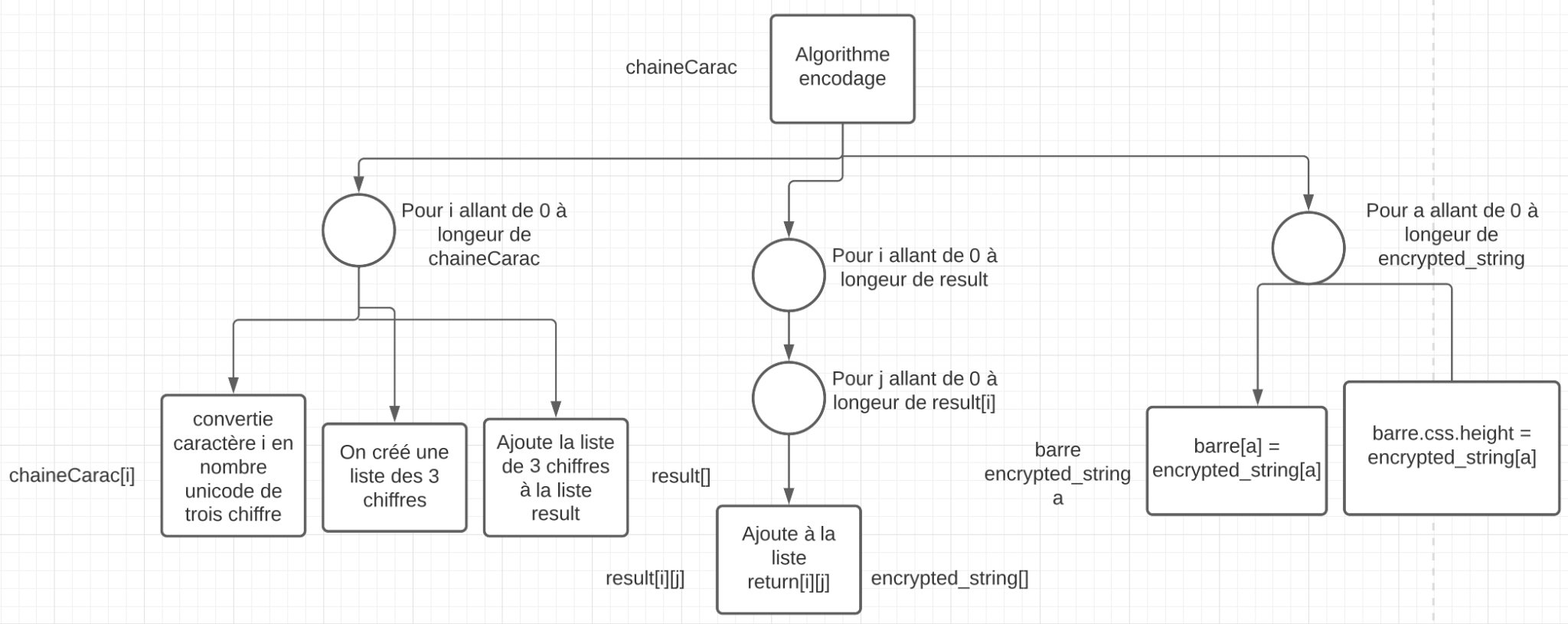




On remarque le logo de l’application ainsi que deux barres la première et la dernière qui servent de marqueurs de début et de fin d’encodage mais également de ratio de comparaison avec les autres barres en faisant : taille du logo / première barre = ratio de comparaison.

Nous prenons que le numéro de licence à encoder pour commencer

On obtient donc pour l’exemple en donnée le numéro de licence suivant : VT0410532



En appliquant l’algorithme suivant :

On obtient :



On a choisi de mettre les barres les plus grandes au début et à la fin pour faire un ratio avec le logo.

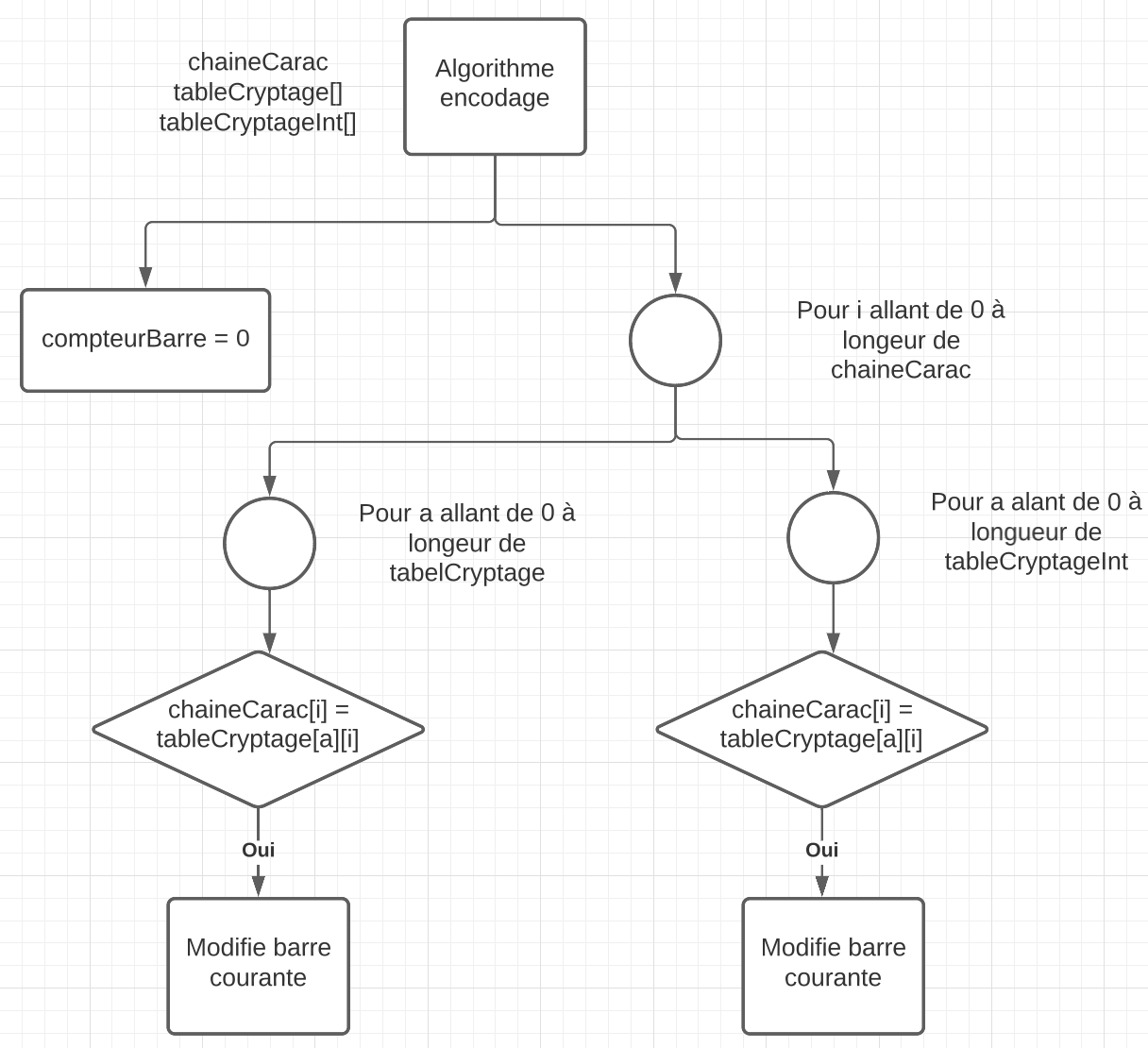
Problème d’unicode : il comprend beaucoup trop de caractère et n’est pas optimisé à notre exemple.

On va donc utiliser la base 36 qui comprend les 26 lettres de l'alphabet (a-z) et les 10 chiffres (0-9). On créé notre propre table de cryptage

| a | 1,1 |
| --- | --- |
| b | 1,2 |
| c | 1,3 |
| d | 1,4 |
| e | 1,5 |
| f | 1,6 |
| g | 1,7 |
| h | 1,8 |
| i | 1,9 |
| j | 2,0 |
| k | 2,1 |
| l | 2,2 |
| m | 2,3 |
| n | 2,4 |
| o | 2,5 |
| p | 2,6 |
| q | 2,7 |
| r | 2,8 |
| s | 2,9 |
| t | 3,0 |

| u | 3,1 |
| --- | --- |
| v | 3,2 |
| w | 3,3 |
| x | 3,4 |
| y | 3,5 |
| z | 3,6 |
| 0 | 3,7 |
| 1 | 3,8 |
| 2 | 3,9 |
| 3 | 4,0 |
| 4 | 4,1 |
| 5 | 4,2 |
| 6 | 4,3 |
| 7 | 4,4 |
| 8 | 4,5 |
| 9 | 4,6 |

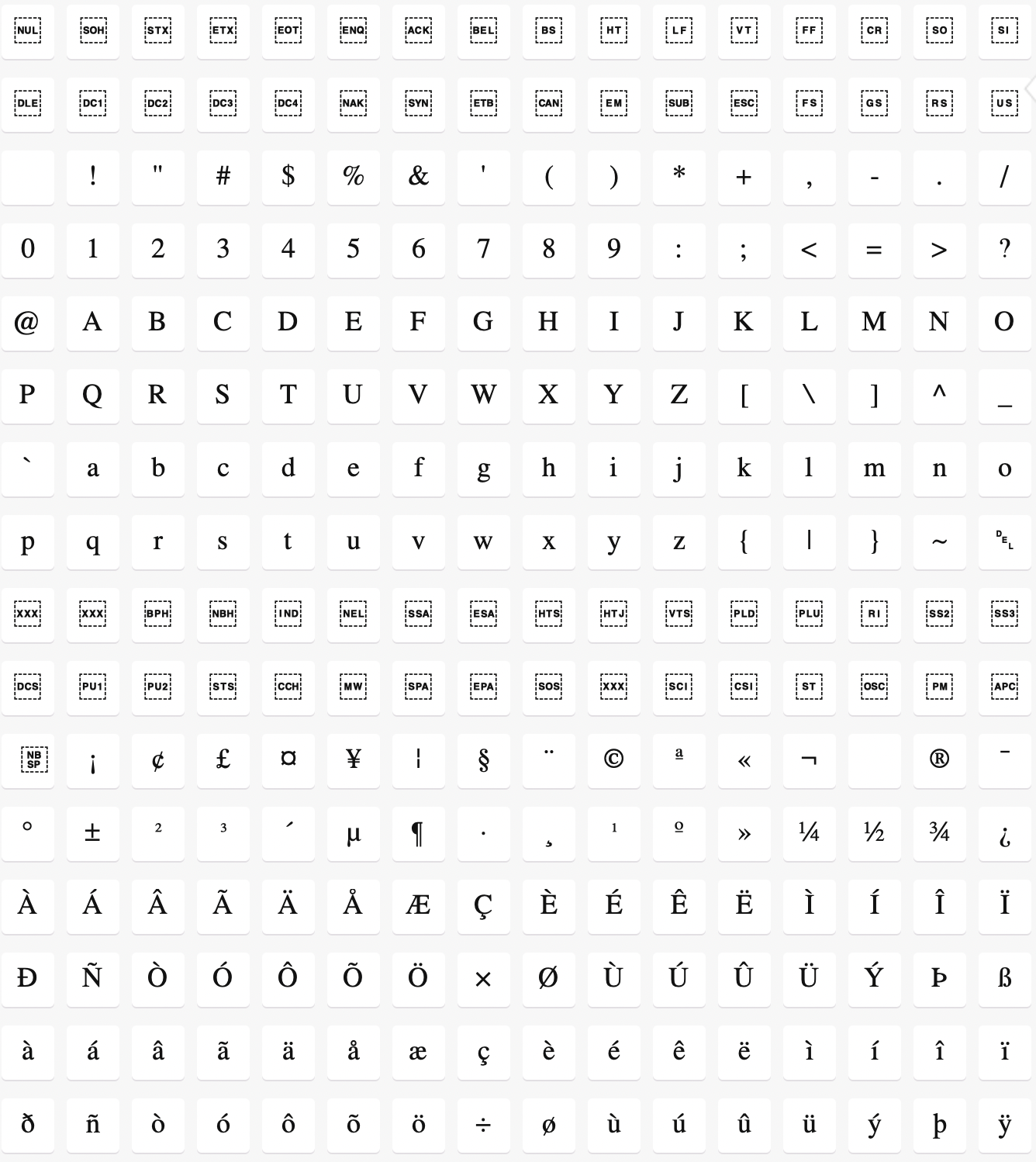
Algorithme encodage en base 36 avec table de cryptage :



Dictionnaire des données

| Nom | Type | Signification |
| --- | --- | --- |
| chaineCarac | chaine de caractère | N° Licence |
| tableCryptage | tableau de caractères | liste contenant chaque lettre de l’alphabet et leur correspondant en encodage en base 36 |
| tableCryptageInt | tableau de caractères | liste contenant chaque chiffre (0 à 9) et leur correspondant en encodage en base 36 |
| compteurBarre | entier non signé | indicateur de la barre en cours de modification |

Unicode :



<https://levelup.gitconnected.com/spotify-codes-and-how-they-work-664f4e4b8489>

<https://boonepeter.github.io/posts/2020-11-10-spotify-codes/>

Skimage

regionprops

Gestion et correction des erreurs

2 types de correction d’erreur:

Codes convolutionnels et blocs de codes linéaires

| Codes convolutionnels | Codes de blocs linéaires |
| --- | --- |
| exemples:  Code Turbo  Diagrammes Trellis | exemples:  codes de Hamming  codes cycliques |

Dans notre cas nous utiliserons un code convolutionnels car la longueur du mot encodé nous importe (pour réduire le nombre de barres)



S3.01 - Développement d’application et Gestion de Projet

Enseignant tuteur : Nathalie VALLES-PARLANGEAU

Groupe n°1

**SAE du Troisième Semestre**

- Pôle Développement -

Spécifications Externes de notre problème algorithmique



Sport Track

Application Web Responsive de gestion de clubs sportifs amateurs

Matis Chabanat | Titouan Cocheril

Arthur Le Menn | Ivan Salle

- - -

TD1 | TP1

BUT Informatique

Semestre 3

- 2022 | 2023 -

Rappel du pitch de notre application

Dans le cadre de ce projet, nous avons décidé d’étudier le thème du sport et plus particulièrement de la gestion du sport amateur.

On retrouve sur internet beaucoup d’applications concernant le sport professionnel qui proposent différentes fonctionnalités telles que la visualisation des résultats ou de la composition des équipes, etc. Mais on en trouve très peu concernant le sport amateur. C’est dans le but de rendre la gestion des clubs sportifs amateur plus simple que nous avons décidé de créer “Sport Track” l’application qui centralise toute la vie du club au même endroit.

Nous souhaitons proposer un produit qui permettra de faire vivre le sport amateur en le rendant accessible à tout sportif licencié dans un club Français.

L’application prendra la forme d’un site web-responsive, c’est-à-dire que l’on pourra y accéder sur n’importe quel navigateur web et donc sur n’importe quel appareil (smartphone, ordinateur). Le produit sera également gratuit car le but est de rendre le sport amateur accessible et visible à tous.

Enfin, nous avons misé sur un style simple et épuré ce qui permettra d’éviter de perdre l’utilisateur avec des interfaces trop complexes.

Données manipulées



**Description des tables :**

Joueurs : Tous les joueurs licenciés

Equipes : Toutes les équipes déclarées

Championnats : Tous les championnats

Matchs : Tous les matchs

Stade : Les stades dans lesquels ont lieu les matchs et entraînements.

evenementsMatch : Les évènements qui ont lieux durant le match (but, carton etc)

jouerMatch : lien des joueurs et des matchs

participerChampionnat : lien entre équipe et championnat.

Messages : Les messages

Entrainement - Autre évènements : Tous les évènements internes des clubs

Inscrit : Tous les joueurs licenciés inscrits

**Problème Algorithmique : Générateur de QR Code**

Pitch du problème

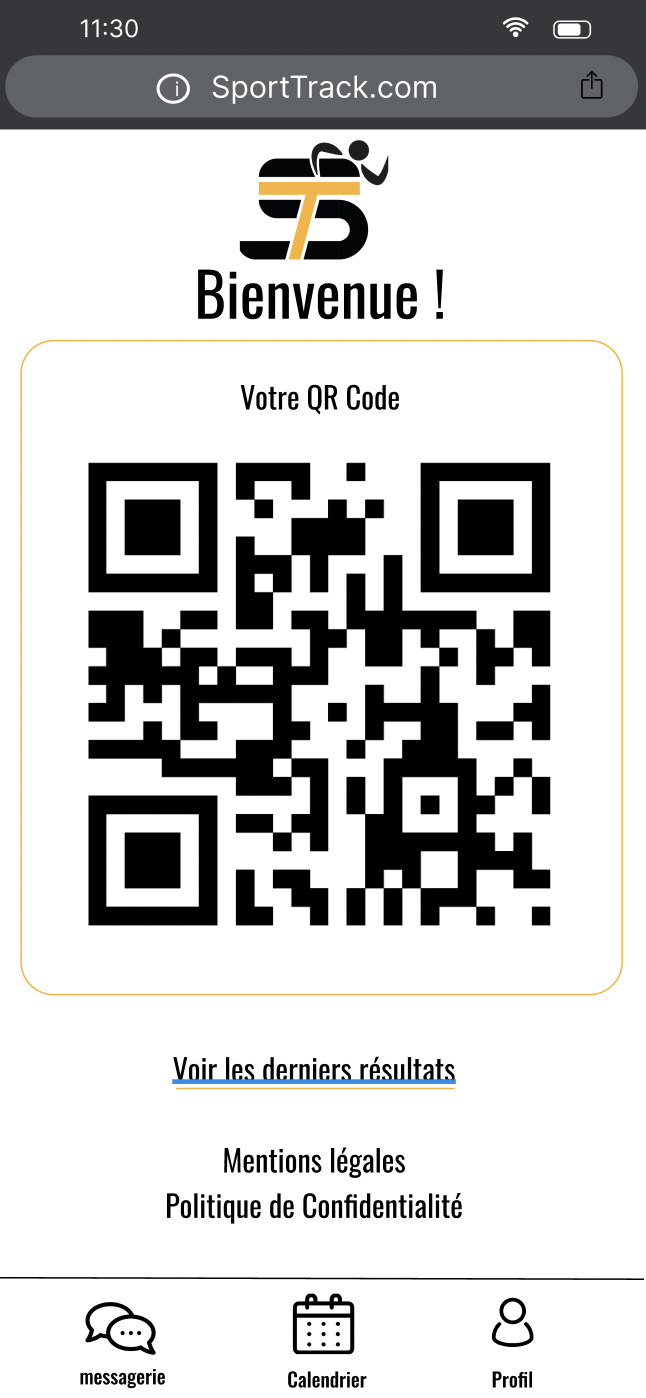
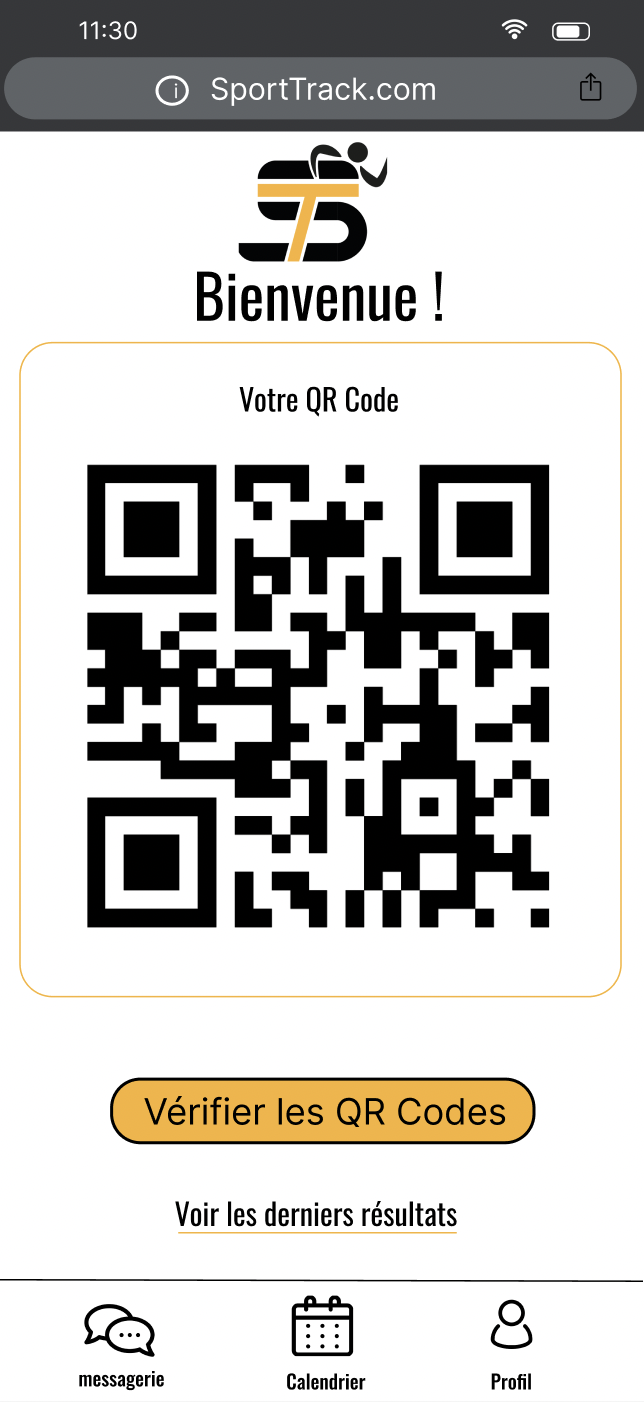
Sur la page d’accueil le joueur trouvera un QR code qu’il pourra venir présenter à l'entraînement et au match. Pour signifier sa présence et pour prouver que c’est bien le joueur possédant la licence et donc la vraie personne.

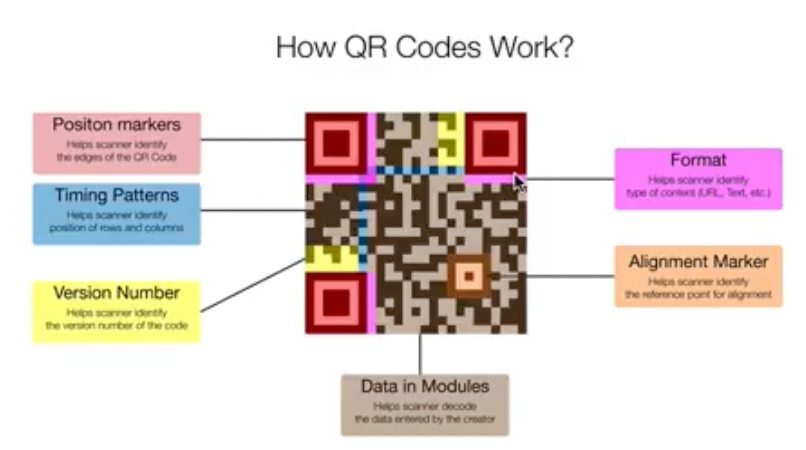
L'entraîneur trouvera sur sa page d’accueil un bouton de vérification de présence de ses joueurs qui accède à l’appareil photo et peut scanner. Il trouvera également son QR code pour signifier sa présence aux matchs.

L’idée Algorithmique : à partir des licences (données) joueurs, entraîneurs générer des QR code (résultats) et pouvoir les vérifier via l’appareil photo (une vérification entre le message retourné par le QR code et une base de données de toutes les licences sera effectuée).



Maquettes / schémas illustrant le problème



Voir annexe pour comprendre comment fonctionne un QR code

Informations manipulées pour traiter le problème

Il existe deux types de QR code statique et dynamique en fonction de si on veut ou non modifier les données entrées. Nos données entrées seront les licences de joueurs et entraîneur, nôtre QR code sera donc statique.

Le résultat sera un QR code généré par la licence.

Une fois scanné le QR code modifie la base de données des présences dans l’équipe, affiche sur l’écran du scanner le succès du scan

Suite de réflexion :

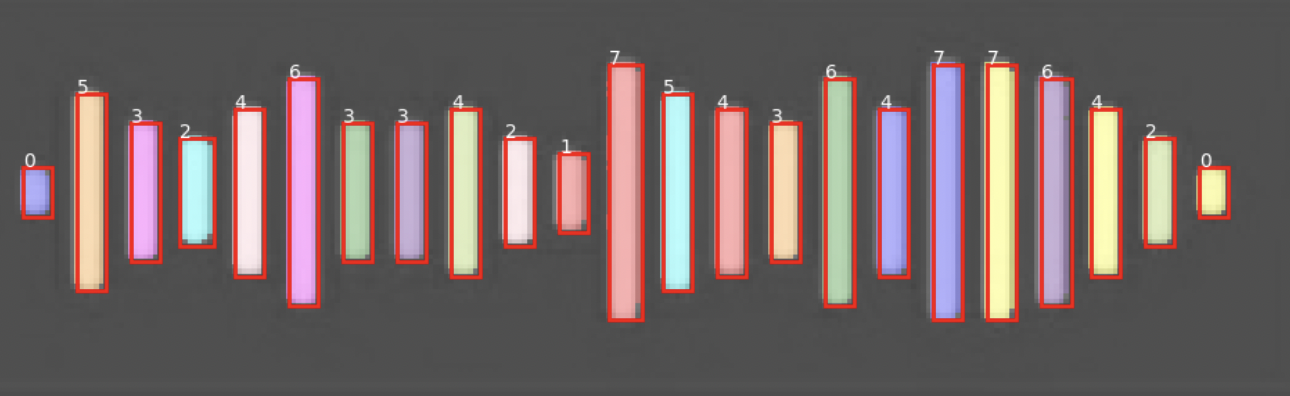
Le QR code était notre première idée pour résoudre le problème algorithmique mais il a néanmoins des avantages/défauts comme le fait qu’il puisse être scanné par tous les appareils, c’est pratique car pas besoin de développer un scanner mais ça pose un problème de sécurité à notre application.

En effet , n'importe qui de non inscrit/connecté à l’application peut obtenir les informations détenu dans le QR code.

On a donc choisi d’encoder nos informations dans un code barres (type spotify) et de développer notre propre scanner.

Comment ça marche :

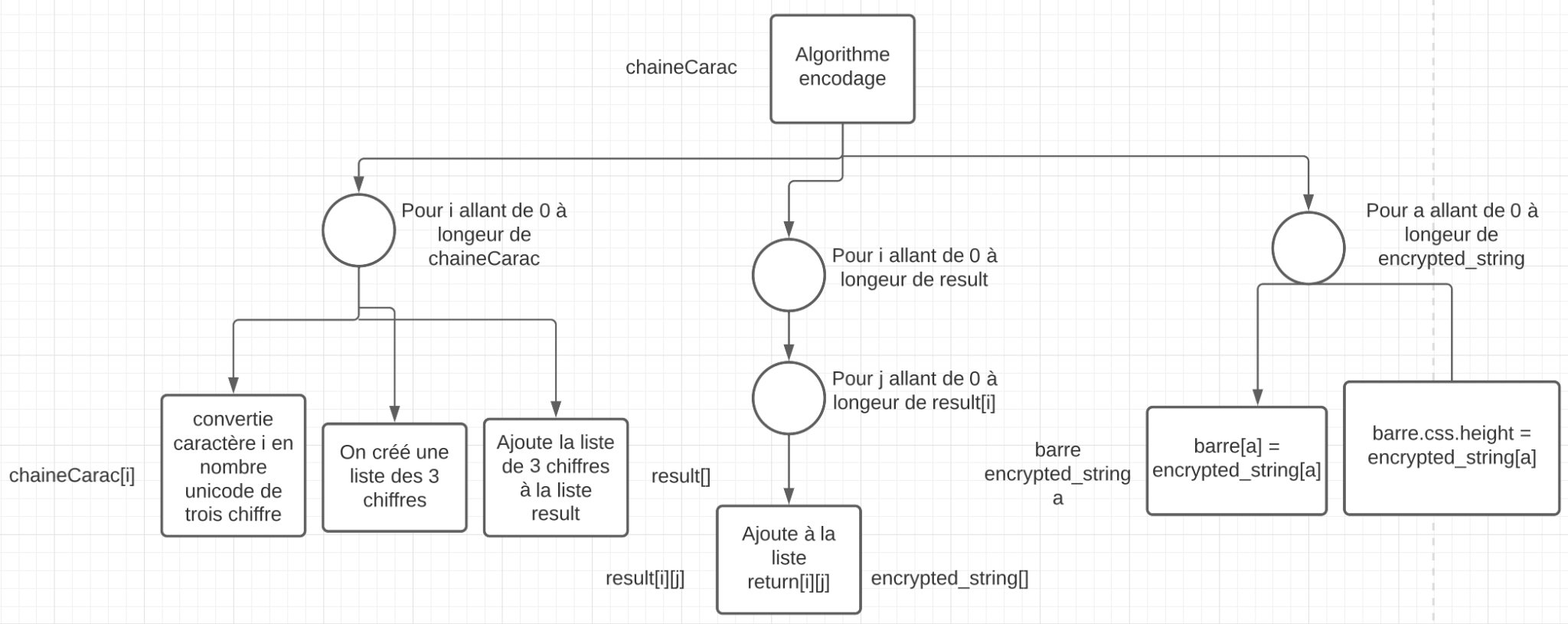




On remarque le logo de l’application ainsi que deux barres la première et la dernière qui servent de marqueurs de début et de fin d’encodage mais également de ratio de comparaison avec les autres barres en faisant : taille du logo / première barre = ratio de comparaison.

Nous prenons que le numéro de licence à encoder pour commencer

On obtient donc pour l’exemple en donnée le numéro de licence suivant : VT0410532



En appliquant l’algorithme suivant :

On obtient :



On a choisi de mettre les barres les plus grandes au début et à la fin pour faire un ratio avec le logo.

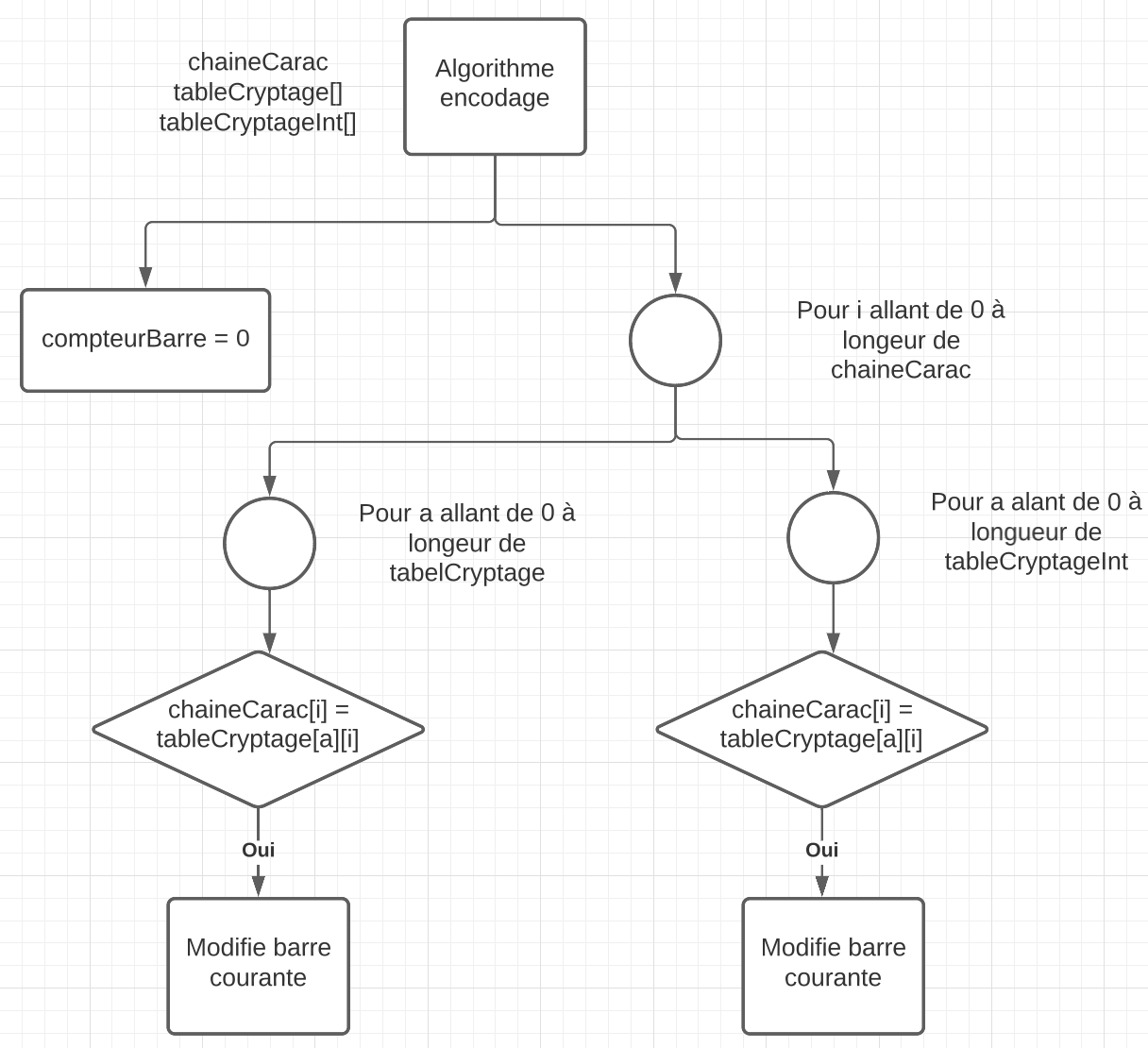
Problème d’unicode : il comprend beaucoup trop de caractère et n’est pas optimisé à notre exemple.

On va donc utiliser la base 36 qui comprend les 26 lettres de l'alphabet (a-z) et les 10 chiffres (0-9). On créé notre propre table de cryptage

| a | 1,1 |
| --- | --- |
| b | 1,2 |
| c | 1,3 |
| d | 1,4 |
| e | 1,5 |
| f | 1,6 |
| g | 1,7 |
| h | 1,8 |
| i | 1,9 |
| j | 2,0 |
| k | 2,1 |
| l | 2,2 |
| m | 2,3 |
| n | 2,4 |
| o | 2,5 |
| p | 2,6 |
| q | 2,7 |
| r | 2,8 |
| s | 2,9 |
| t | 3,0 |

| u | 3,1 |
| --- | --- |
| v | 3,2 |
| w | 3,3 |
| x | 3,4 |
| y | 3,5 |
| z | 3,6 |
| 0 | 3,7 |
| 1 | 3,8 |
| 2 | 3,9 |
| 3 | 4,0 |
| 4 | 4,1 |
| 5 | 4,2 |
| 6 | 4,3 |
| 7 | 4,4 |
| 8 | 4,5 |
| 9 | 4,6 |

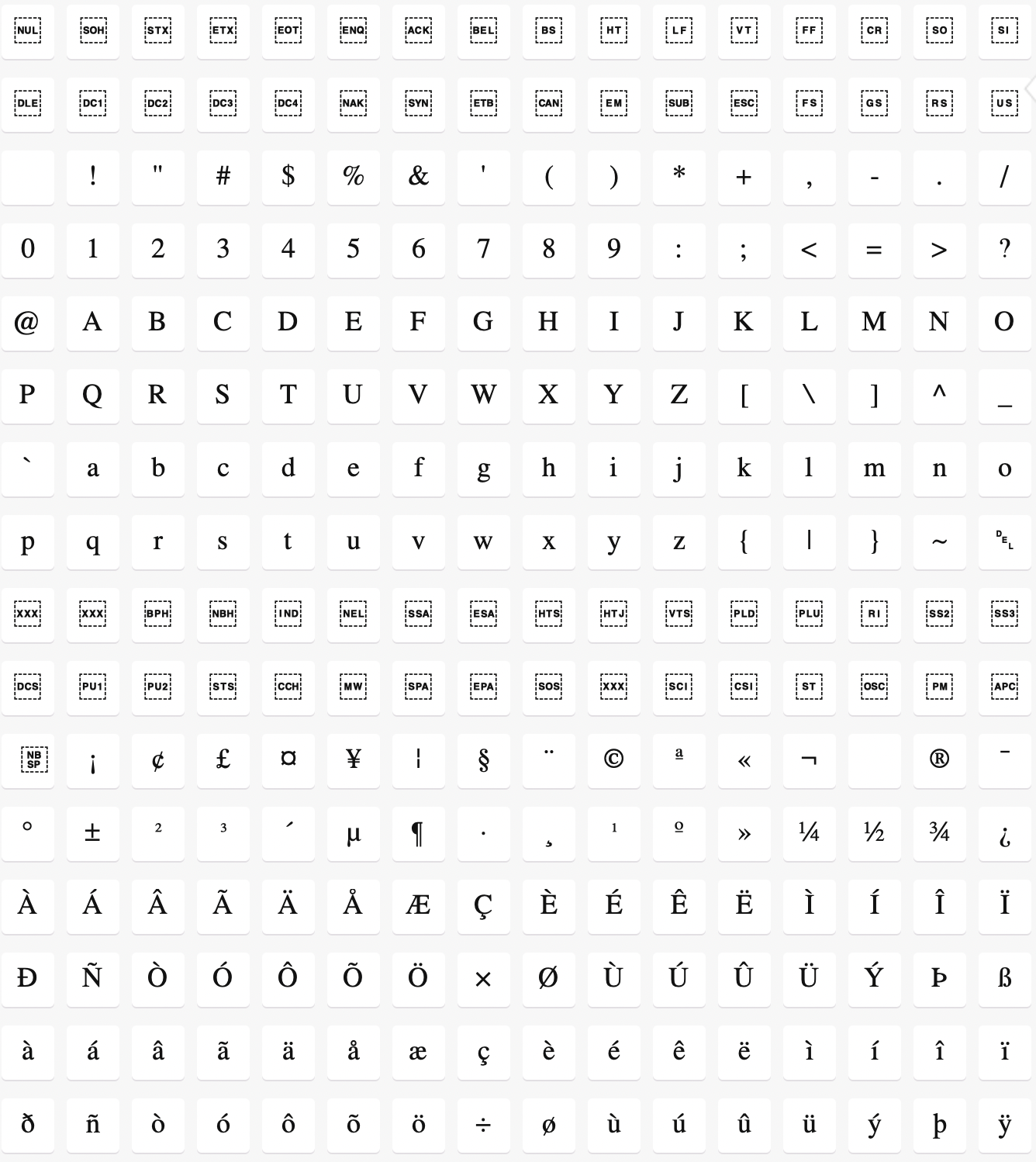
Algorithme encodage en base 36 avec table de cryptage :



Dictionnaire des données

| Nom | Type | Signification |
| --- | --- | --- |
| chaineCarac | chaine de caractère | N° Licence |
| tableCryptage | tableau de caractères | liste contenant chaque lettre de l’alphabet et leur correspondant en encodage en base 36 |
| tableCryptageInt | tableau de caractères | liste contenant chaque chiffre (0 à 9) et leur correspondant en encodage en base 36 |
| compteurBarre | entier non signé | indicateur de la barre en cours de modification |

Unicode :



<https://levelup.gitconnected.com/spotify-codes-and-how-they-work-664f4e4b8489>

<https://boonepeter.github.io/posts/2020-11-10-spotify-codes/>

Skimage

regionprops

Gestion et correction des erreurs

2 types de correction d’erreur:

Codes convolutionnels et blocs de codes linéaires

| Codes convolutionnels | Codes de blocs linéaires |
| --- | --- |
| exemples:  Code Turbo  Diagrammes Trellis | exemples:  codes de Hamming  codes cycliques |

Dans notre cas nous utiliserons un code convolutionnels car la longueur du mot encodé nous importe (pour réduire le nombre de barres)

GRAY CODE sur 6 bits (62 mots différents donc 2^6)

gray code : comme le binaire mais 1 seul bit change entre chaque caractère suivant pour faciliter la correction des erreurs.

| a | 000000 |
| --- | --- |
| b | 000001 |
| c | 000011 |
| d | 000111 |
| e | 001111 |
| f | 011111 |
| g | 111111 |
| h | 111110 |
| i | 111100 |
| j | 111000 |
| k | 110000 |
| l | 100000 |
| m | 100001 |
| n | 100011 |
| o | 100111 |
| p | 101111 |
| q | 101110 |
| r | 101100 |
| s | 101000 |
| t | 001000 |

| u | 001100 |
| --- | --- |
| v | 011100 |
| w | 011110 |
| x | 010110 |
| y | 010100 |
| z | 010101 |
| 0 | 010101 |
| 1 | 011101 |
| 2 | 001101 |
| 3 | 001001 |
| 4 | 000001 |
| 5 | 101001 |
| 6 | 101011 |
| 7 | 111011 |
| 8 | 111010 |
| 9 | 101010 |