**Laboratoire #4**

Stockage local, internationalinalisanalisationnement et requête avec authentification

**Étape 1 - La routine 📦**

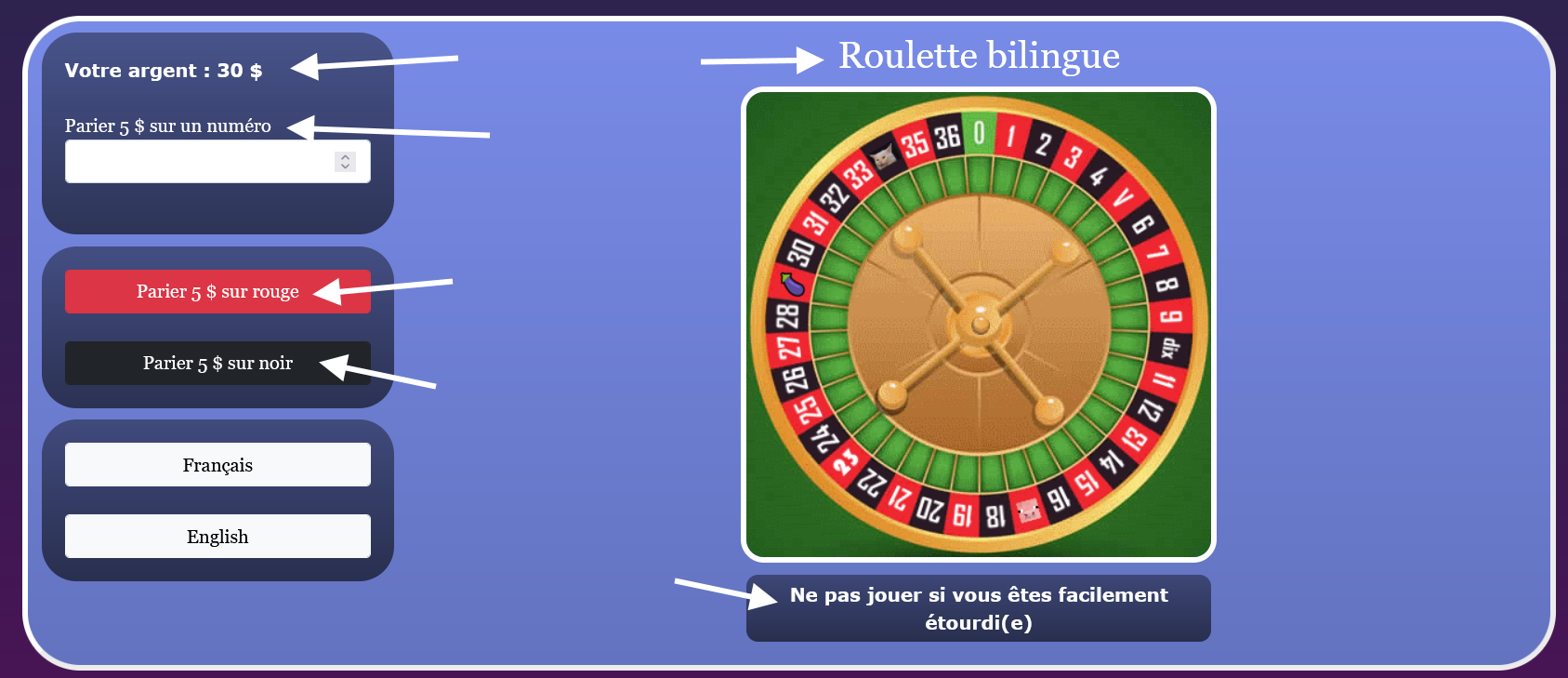
* Réinstallez les dépendances du projet **Angular✨** fourni à l’aide de PowerShell.
  + **10**  De plus, ajoutez les **dépendances** qui seront nécessaires pour **internationaliser** l’application Web. 👅
* Ouvrez le projet avec Visual Studio Code et exécutez l’application Web.

**Étape 2 - À la conquête de la persistance ⌚⏳**

* Pour l’étape 2, nous allons travailler sur le composant **stockage**. (« Machine à sous »)
* Commencez par vous familiariser avec son fonctionnement :
  + Créez un **profil** dans la page.
  + **Jouez** un peu et remarquez que votre portefeuille évolue.
  + Quittez ce composant (à l’aide du **menu de navigation**) et revenez-y.
    - Vous remarquerez alors que votre profil n’existe plus... et que vous devrez en créer un nouveau ! En effet, toute trace de vos données précédentes a été effacée. Notre mission sera de rendre votre profil persistant.
* **3-9**  Faites le nécessaire pour ranger le profil dans le **stockage de session**.
  + Le **profil** doit être rangé (et mis à jour) lors de sa **création** ET à chaque fois que la valeur de votre portefeuille évolue.
  + Le **profil** doit être **chargé** (récupéré dans le **stockage de session**) à chaque fois que le **composant** est chargé. (Donc dès que la page est affichée)
    - Notez que la variable de classe **this.profile** sert à stocker, en tout temps, le **profil** du joueur pendant que le **composant** est actif.

**Étape 3 - À la conquête des langues 👅👄**

* Pour l’étape 3, nous allons travailler sur le composant **i18n**. (« Roulette bilingue »)
* **10-17**  Internationalisez le texte indiqué dans l’image ci-dessous dans le template HTML.
  + Vous avez déjà installé les dépendances nécessaires, mais il faudra...
    - Configurer l’internationalisation dans **app.config.ts**.
    - Injecter le **service** nécessaire dans le composant **i18n**.
    - Définir la langue par défaut. (Français)
    - Étiqueter les contenus internationalisés. Attention, pour l’argent, une variable est mêlée au texte à traduire …
    - Produire les fichiers de traduction.
    - Rendre fonctionnels les boutons « Français » et « English ».
  + Étant donné que l’anglais est un dialecte mystérieux qui vous est inconnu, tous les textes vous sont fournis sous l’image ci-dessous.



**Français**

Votre argent : X $

• Parier 5 $ sur un numéro

• Parier 5 $ sur rouge

• Parier 5 $ sur noir

• Roulette bilingue

• Ne pas jouer si vous êtes facilement étourdi(e)

**Anglais**

• Your money : $ X

• Bet $ 5 on a number

• Bet $ 5 on red

• Bet $ 5 on black

• Bilingual roulette

• Do not play if you’re easily dizzy

**Étape 4 - Un serveur qui ne répond pas à n’importe qui ✋🛑🎵**

Cette étape est dangereusement facile ⚠ et sert surtout à vous offrir un exemple de code qui fait une requête à Spotify en vue du TP2.

* Nous utiliserons le composant **token** et le service **Spotify** pour cette étape.
* **18-19**  D’abord, créez-vous un compte Spotify (si vous n’en avez pas) et obtenez un CLIENT ID et un CLIENT SECRET pour les hardcoder dans le service **Spotify**. (Trouvez les deux constantes en haut)
* Initialement, vous remarquerez que la page n’affiche rien. C’est normal. Avant de NE SERAIT-CE permettre à l’utilisateur d’utiliser cette page, on doit s’assurer de s’être « authentifié » à l’API de Spotify.
  + **20**  La fonction de connexion à l’API de Spotify est déjà codée pour vous. (Dans le service Spotify) Trouvez où appeler cette fonction à partir du composant **token** pour qu’un token d’authentification soit obtenu dès le chargement de la page.
  + **21**  Une fonction permettant de rechercher un artiste dans l’API Spotify et d’afficher son nom et sa photo dans la page est déjà présente dans le service Spotify. Trouvez où appeler cette fonction à partir du composant **token** pour que le formulaire de recherche soit fonctionnel.