**Shakhmat**

Projet d’apprentissage des ouvertures aux échecs

# Historique des versions

|  |  |  |
| --- | --- | --- |
| Numéro de version | Modifications | Commentaires |
| 0.1 | Initialisation du document |  |

Table des matières

[Historique des versions 2](#_Toc103676161)

[Numéro de version 2](#_Toc103676162)

[Modifications 2](#_Toc103676163)

[Commentaires 2](#_Toc103676164)

[1) Présentation et objectif 4](#_Toc103676165)

[2) Maquettes 4](#_Toc103676166)

[3) Liste de taches 5](#_Toc103676167)

[Ecran d’accueil 5](#_Toc103676168)

[Ecran d’apprentissage 5](#_Toc103676169)

[Exemple d’ajout d’ouvertures : 5](#_Toc103676170)

[Modules 6](#_Toc103676171)

[Module front 6](#_Toc103676172)

[Module back 6](#_Toc103676173)

# Présentation et objectif

Le présent document servira de documentation pour le projet Shakhmat (jeu d’échecs en arménien). Avec sa bibliothèque d’ouvertures, l’application permettra d’apprendre quelques ouvertures.

# Maquettes

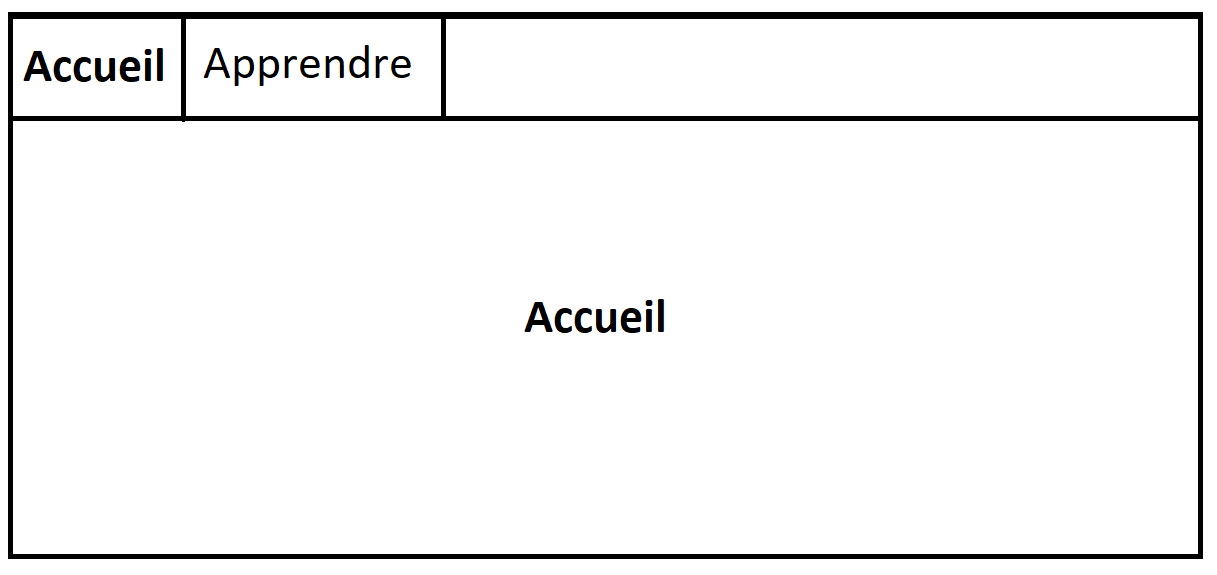


Figure 1 : Page d'accueil

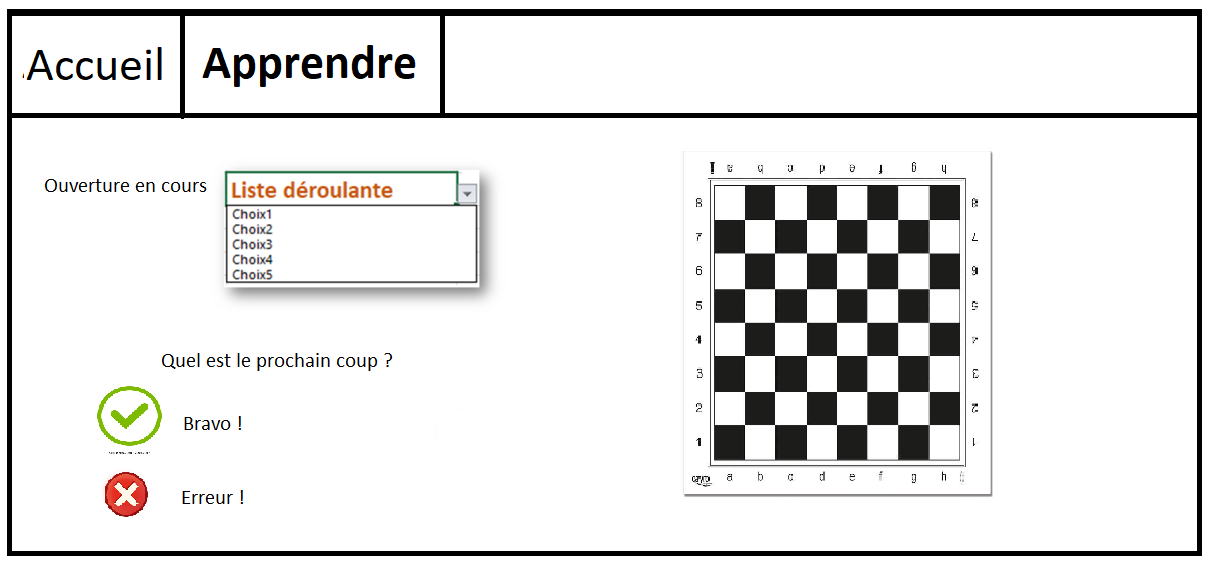


Figure 2 : Page d'apprentissage

# Liste de taches

## Ecran d’accueil

Page d’accueil, permettant d’accéder aux différents onglets de l’application.

### Ecran d’apprentissage

Cet écran contient

* Une liste déroulante avec les différentes ouvertures en base de donnée. Au clic sur une ouverture, les données la concernant seront chargées. Appel d’un service qui va remonter les informations de l’ouverture.
* Un échiquier qui retranscrit les coups. A chaque coup, l’échiquier sera mis a à jour
* Un champ permettant à l’utilisateur d’écrire le prochain coup de l’ouverture. Ce champ sera lu et interprété pour que le coup soit joué sur l’échiquier
* Front
  + Création de l’écran et ajout de l’onglet
* Réflexion sur comment utiliser l’echiquier en javascript
  + Introduction dans le projet
* Choix de l’ouverture
  + Appel en base pour récupérer les données de l’ouverture => appel REST
* Encart pour entrer les coups
  + Contrôle dans le front pour les coups
  + Appel du chessboard
  + Si contrôle KO, message d’erreur
  + Si contrôle ok, ça joue le coup
  + Une fois le coup joué, coup suivant demandé
* Pop up d’aide ?
  + Au click sur aide, un indice sur la pièce à jouer s’affiche
* Choix d’une nouvelle ouverture
  + Lors de la « session » on peut choisir une autre ouverture
  + A la fin d’une session, on peut soit recommencer la session soit choisir une autre ouverture

## Exemple d’ajout d’ouvertures :

* Un écran d’ajout d’ouvertures :
  + Cet écran contiendrait un formulaire qui contiendrait les éléments suivants :
    - Nom de l’ouverture
    - Liste des coups dans le bon format
    - Ou : lecture d’un fichier qui contiendrait ces éléments qui serait chargé
  + Une fois complété dans le bon format, l’application enregistrerait les données en base => création d’un fichier pour les lire et les proposer dans l’application
  + Cet écran pose le problème de coups impossibles lors de l’ajout

## Modules

* Module back
* Module front

### Module front

* 
  + Ce zip contient un module avec échiquier + méthodes classiques pour jouer aux échecs. En lui rentrant des coups, il joue les coups sur l’échiquier. Développé en javascript
* Permet l’affichage en liste déroulante des ouvertures disponibles. Au moment du choix, l’application va remonter les coups de la base de donnée => appel au back
* Contient un champ dans lequel l’utilisateur va rentrer un coup. Le coup va être contrôler selon le module chessboardjs, affiche un message d’erreur si le coup n’est pas bon. S’il est bon, il va jouer le coup en utilisant chessboardjs.
* Si l’ouverture est ok, l’application va afficher un message « Bravo ! » Sinon il affiche un message d’erreur avec un bouton pour ré essayer.

### Module back

* Appel en « base de données » => fichiers représentant les ouvertures