

# Slim Ninja

Documentation technique

Stefan Nikolic - 2023



## Table des matières

1.Table de versions.....	3
2.Introduction.....	3
3.Rappel de l'énoncé.....	3
3.1 Organisation .....	3
3.2 Objectif .....	3
3.3 Spécifications.....	4
3.4 Matériel et logiciels à disposition.....	4
3.5 Livrables.....	5
4.Planification.....	5
4.1 Diagramme de Gantt prévisionnel .....	5
4.2 Diagramme de Gantt finale : .....	6
5.Méthodologie en six étapes .....	6
5.1 S'informer.....	6
5.2 Planifier.....	6
5.3 Décider .....	6
5.4 Réaliser .....	7
5.5 Contrôler .....	7
5.6 Évaluer.....	7
6.Implémentation.....	7
6.1 Base de données.....	7
6.1A Table game : .....	9
6.1B Table user .....	10
6.1C Table item.....	10
6.2 Fonctions PHP.....	11
6.2A Ficher user.php.....	11
6.2B Fichier food.php .....	13
6.2C Ficher conf.php.....	16
6.2D Fichier myDB.php.....	17
6.3 API.....	17
6.3A Fichier getFoods.php.....	18
6.3B Fichier getUsers.php.....	18
6.4 Fonctions Javascript .....	19
6.4A Fichier user.js .....	19
6.4B Fichier game.js.....	19
6.4C Fichier UpdateUsers.js.....	25

6.4D Fichier addBadFood.js.....	25
6.4E Fichier getFoods.js.....	26
6.4F Fichier sendDataToPhp.js .....	26
7.Problèmes rencontrés .....	27
8.Maquettes .....	27
8.1 Page Inscription .....	28
8.2 Page Login.....	28
8.3 Page Index .....	29
8.4 Page Account .....	29
8.5 Page Rules .....	30
8.6 Page Games .....	30
8.7 Page Game.....	31
8.8 Fin d'une partie .....	31
8.9 Schéma des groupes d'aliments.....	32
8.10 Schéma de navigation du site.....	32
9.Plan de tests et rapport de tests .....	33
9.1 Configuration :.....	33
9.2 Tests :.....	33
9.2A Fonctions PHP .....	33
9.2B Fonctions JavaScript.....	39
10.Conclusion .....	45
10.1 Futurs améliorations .....	45
10.2 Bilan personnel.....	45
11.Glossaire .....	46
12.Bibliographie .....	46
12.1 ChatGPT .....	47
13.Remerciement .....	47
14.Code sources .....	48
Annexe 1.....	48

# 1.Table de versions

Version	Nom	Date de début	Date de la fin
1.0.0	Slim-Ninja	27.04.2023	17.05.2023

# 2.Introduction

Ce document permet de présenter un rapport détaillé sur mon travail pratique individuel (TPI). Dans ce travail il m'a été imposé de réaliser un jeu pédagogique sur l'alimentation en 11 jours (88 heures) qui aura pour but de faire apprendre aux joueurs les différents groupes d'aliments.

Slim Ninja est un jeu multijoueur/solo qui permet aux joueurs d'apprendre divers groupes d'aliments tout en jouant. Les joueurs doivent appuyer sur les bons aliments de la catégorie choisie pour gagner des points. Les aliments tombent du haut vers le bas aléatoirement sur l'écran. Si un joueur appuie sur un aliment malsain ou si son score devient négatif il perd la partie. Le joueur qui a le plus grand score de la partie fait défiler les aliments de l'autre joueur plus vite. Le dernier joueur restant gagne la partie.

# 3.Rappel de l'énoncé

## 3.1 Organisation

Élève : Nikolic Stefan [stefan.nklc@eduge.ch](mailto:stefan.nklc@eduge.ch)

Maître : Oscar François [oscar.francois@eduge.ch](mailto:oscar.francois@eduge.ch)

Experts: Borys Folomietow [borys@folomietow.ch](mailto:borys@folomietow.ch)

Vanini Daniel [daniel.vanini@skyguide.ch](mailto:daniel.vanini@skyguide.ch)

## 3.2 Objectif

L'objectif de ce projet est de réaliser en 11 jours (88 heures) un jeu vidéo à but pédagogique en Javascript et en PHP. Le jeu permet aux utilisateurs d'apprendre les groupes d'aliments de manière ludique. Les aliments tombent verticalement de manière aléatoire. Les scores des utilisateurs augmentent lorsque les utilisateurs appuient sur les bons aliments et diminuent s'ils appuient sur les aliments qui ne correspondent pas à la catégorie choisie. Si un joueur appuie sur un des aliments malsains ou si son score est négatif, il perd la partie. Le dernier joueur restant gagne la partie.

### 3.3 Spécifications

**Les points principaux du jeu sont les suivant :**

- Le jeu est en 2 dimensions
- Le joueur peut choisir parmi 5 catégories d'aliments (fruit, légumes, légumineuses, céréales et féculant).
- Les aliments tombent verticalement de manière aléatoire. Leur positionnement horizontal est aussi aléatoire.
- Le nombre d'aliments et la vitesse du défilement augmente progressivement pendant la partie.
- Chaque aliment est affiché avec une image et du texte le décrivant.
- Lorsque l'utilisateur appuie sur les bons aliments correspondant à la catégorie du jeu choisi, son score augmente.
- Le score diminue si l'utilisateur sélectionne un aliment de la mauvaise catégorie, mais aussi s'il ne sélectionne pas le bon aliment avant qu'il n'atteigne le bas de l'écran.
- Des aliments malsains apparaissent aléatoirement. Si l'utilisateur en sélectionne un, le jeu s'arrête.
- Si le score du joueur est négatif il perd la partie.
- Un joueur sur un ordinateur distant peut rejoindre la partie déjà en cours.
- En multijoueur le joueur qui a le score le plus élevé fait augmenter la vitesse du défilement de l'autre joueur.
- Si un joueur quitte une partie multijoueur, il ne peut plus la rejoindre jusqu'à la fin de cette dernière.
- Les données des joueurs sont systématiquement affichées sur l'écran (nom, score, vitesse du défilement et le temps écoulé depuis le début de la partie)
- Le joueur peut créer et supprimer son compte.

### 3.4 Matériel et logiciels à disposition

**Matériel physique :**

- Pour le TPI, j'utilise un ordinateur qui possède un écran.
- Je possède aussi un journal de bord papier.

**Pour réaliser les différents éléments de ma documentation technique et de mon jeu j'ai utilisé les logiciels suivants :**

- **Excel** pour le diagramme de Gantt
- **Gimp** pour détouurer les images
- **Word** pour la documentation technique et le manuel utilisateur.
- **Wamp-server** pour le serveur.
- **Bloc-notes** pour réaliser les maquettes.
- **Visual studio code** pour éditer le code.
- **GITHUB Desktop** pour versionner mon projet.

## 3.5 Livrables

**À la fin du TPI, les éléments suivants sont rendus aux experts et au maître de TPI sous format électronique :**

- La documentation technique + code source (PDF)
- Manuel utilisateur (PDF)
- Résumé du rapport de TPI (PDF)

**Le maître du TPI recevra, en plus, les éléments suivants :**

- Le carnet de bord
- Le code source

Le projet est disponible à l'adresse suivante :

<https://github.com/stefan1217/TPi-main>

## 4. Planification

### 4.1 Diagramme de Gantt prévisionnel

Voici mon diagramme de Gantt prévisionnel, vous pouvez y trouver toutes les tâches que j'ai définies pour mon projet avec le temps nécessaire estimé.

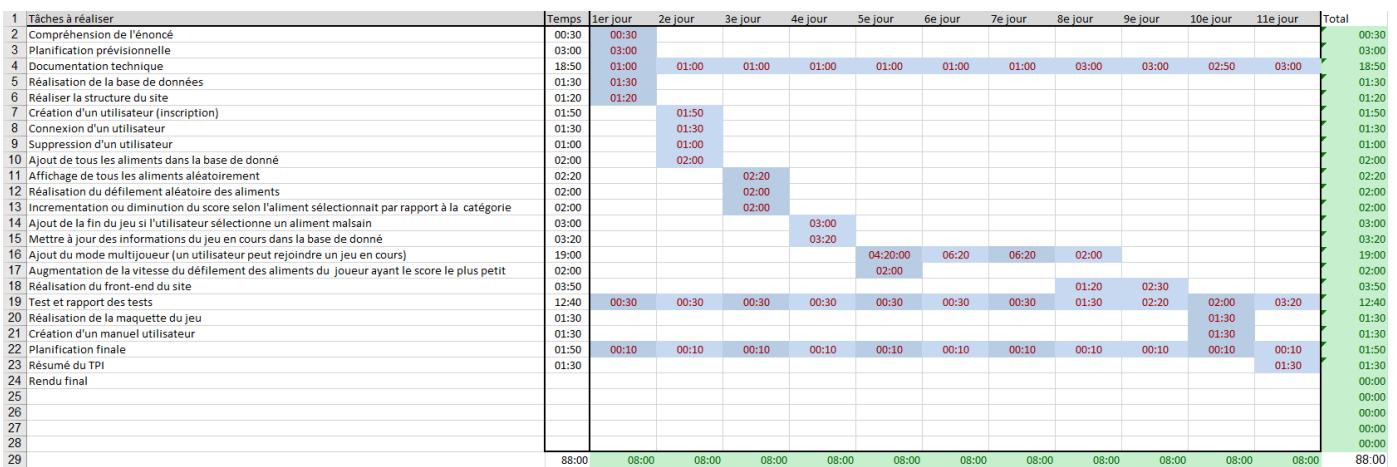


Figure 1 diagramme de Gantt prévisionnel

## 4.2 Diagramme de Gantt finale :

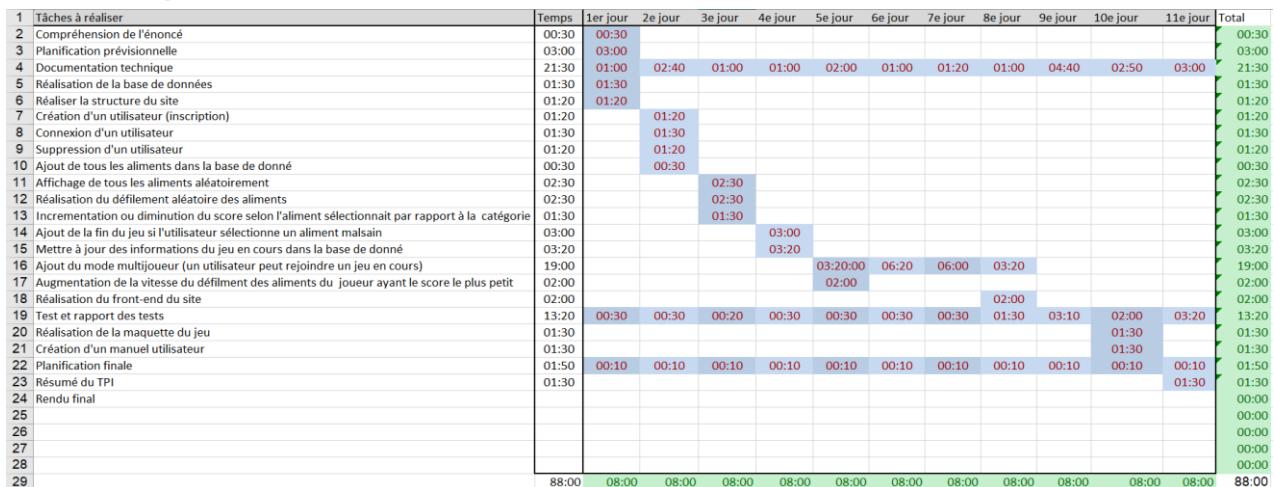


Figure 2 digramme de Gantt finale

Comme on peut le remarquer, le diagramme de Gantt a été respecté dans l'ensemble. Pour certaines tâches, j'ai surestimé le temps, ce qui m'a permis d'avancer sur d'autres tâches. Vers la fin du projet, j'avais terminé le code en avance et j'ai pu, grâce à cet avancement, progresser sur ma documentation technique.

## 5. Méthodologie en six étapes

Pour ce travail, j'ai choisi d'utiliser la méthodologie en six étapes. C'est une méthodologie qui est rapide à mettre en place, donc elle est très utile pour ce projet.

### 5.1 S'informer

Tout au début de mon TPI j'ai dû bien prendre connaissance de mon énoncé pour pouvoir bien comprendre toutes les parties importantes à réaliser ainsi que demander à mon maître de TPI toutes les choses pour lesquelles je n'étais pas sûr d'avoir bien compris pour être bien au clair sur ce qui a été demandé.

### 5.2 Planifier

Deuxièmement, j'ai dû planifier mon travail à l'aide d'un diagramme de Gantt qui aura pour but de m'aider et me guider à réaliser toutes les tâches de ce travail.

### 5.3 Décider

Au cours de mon travail j'ai dû prendre de multiples décisions car le diagramme de Gantt prévisionnel ne prenait pas toutes les tâches en compte et certaines de ces tâches prenaient plus de temps à réaliser que prévu.

## 5.4 Réaliser

Après avoir choisie de réaliser une tâche j'effectue le travail que je pense être en mesure de faire que ce soit pour adapter mon code ou pour réaliser la documentation technique.

## 5.5 Contrôler

Une fois que je terminais une nouvelle fonction, tâche je prenais du temps pour effectuer une multitude de tests qui ont comme objectif de corriger des éventuels bugs, problèmes d'optimisation, failles techniques.

## 5.6 Évaluer

Pour finir, la dernière étape de la méthodologie en six étapes est une évaluation générale du travail que j'ai réalisé et accompli ou échoué à accomplir, j'ai écrit dans mon journal de bord chaque jour toutes les tâches que je réalisais et comment je m'y prenais pour les réaliser.

# 6.Implémentation

## 6.1 Base de données

### Model conceptuel donné :

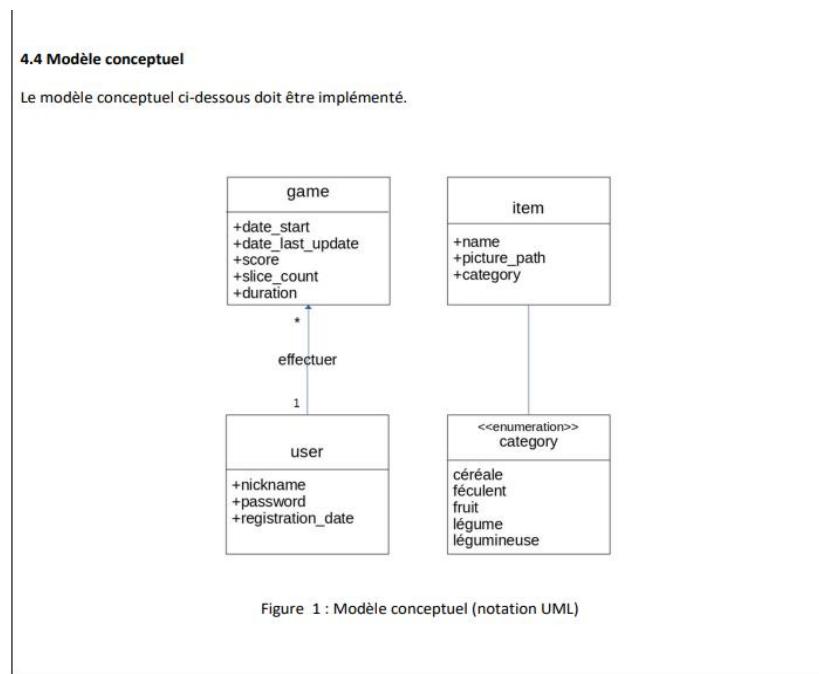


Figure 3 MCD

Dans l'énoncé de mon TPI le model conceptuel de données (MCD) m'a été donné dans le document. J'ai décidé de changer cd model conceptuel après une consultation avec mon prof de TPI pour des raisons fonctionnelles.

**Voici la version finale après modification :**

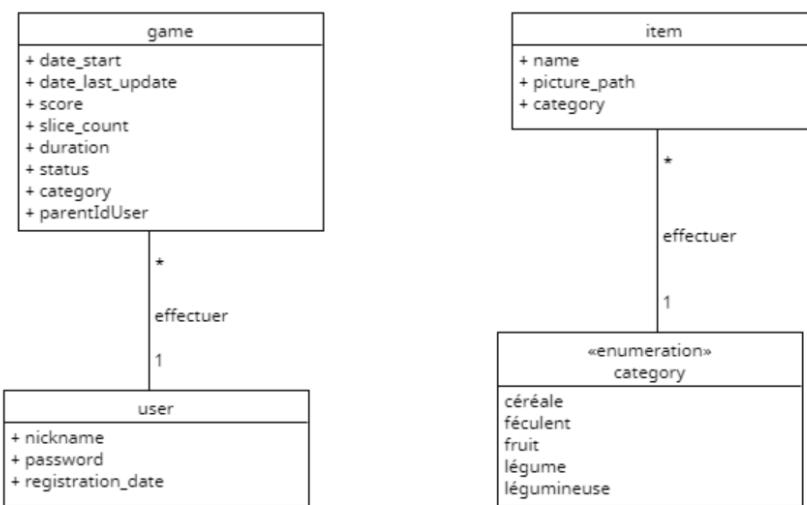


Figure 4 MCD après modification

Le modèle logique de données (MLD) :

**Table game (idGame, date\_start, date\_last\_update, score, slice\_count, duration, status, category, parentIdUser , #idUser)**

**Table user (idUser, nickname, password, registration\_date)**

**Table item (idItem, name, picture\_path, category)**

**Champ category enum (céréal, féculent, fruit, légume, légumineuse)**

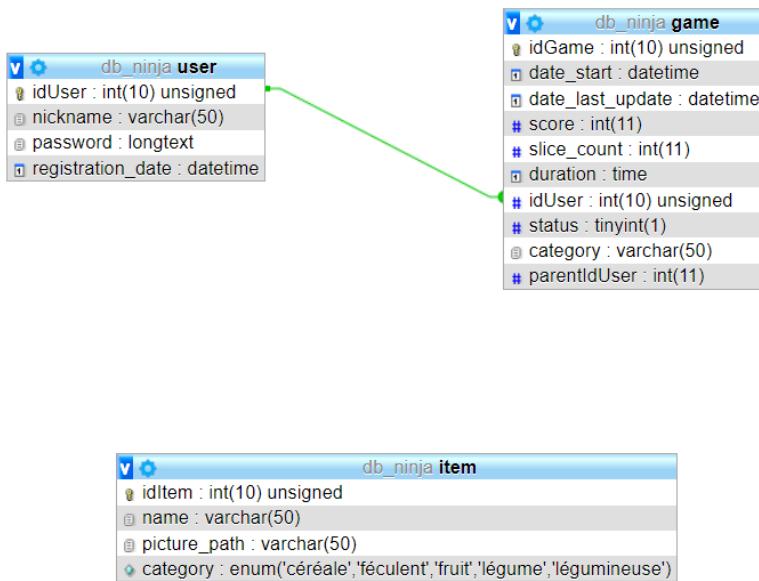


Figure 5 Modèle vue de la BD

## 6.1A Table game :

Nom	Type	Non null	Auto-Incrément	Clé
idGame	Int (10)	Oui	Oui	Primaire
date_start	datetime	Oui	Non	Aucune
date_last_update	datetime	Oui	Non	Aucune
score	Int (11)	Oui	Non	Aucune
slice_count	Int (11)	Oui	Non	Aucune
duration	time	Oui	Non	Aucune
idUser	Int (10)	Oui	Non	Étrangère
status	tinyint (1)	Oui	Non	Aucune
category	Varchar (50)	Oui	Non	Aucune
parentIdUser	Int (11)	Oui	Non	Aucune

Dans cette table :

- Le champ « idGame » est la clé primaire
- Le champ « date\_start » permet de stocker la date de début de la partie
- Le champ « date\_last\_update » permet de stocker la date de la dernière mise à jour de la partie
- Le champ « score » permet de stocker le score du joueur
- Le champ « slice\_count » permet de stocker le nombre d'aliments sur lesquelles le joueur a appuyé
- Le champ « duration » permet de stocker la durée de la partie
- Le champ « idUser » est la clé étrangère de la table user
- Le champ « status » est un booléen qui permet de stocker si la partie est finie ou pas
- Le champ « category » permet de stocker la catégorie que le joueur a choisie
- Le champ « parentIdUser » permet de stocker l'id du joueur qui a créé la partie

## 6.1B Table user

Nom	Type	Non null	Auto-Incrément	Clé
idUser	Int (10)	Oui	Oui	Primaire
nickname	Varchar (50)	Oui	Non	Aucune
password	longtext	Oui	Non	Aucune
registration_date	datetime	Oui	Non	Aucune

Dans cette table :

- Le champ « idUser » est la clé primaire
- Le champ « nickname » permet de stocker le pseudo du joueur
- Le champ « password » permet de stocker le mot de passe haché du joueur
- Le champ « registration\_date » permet de stocker la date de la création du compte

## 6.1C Table item

Nom	Type	Non null	Auto-Incrément	Clé
idItem	Int (10)	Oui	Oui	Primaire
name	Varchar (50)	Oui	Non	Aucune
picture_path	Varchar (50)	Oui	Non	Aucune
category	enum	Oui	Non	Aucune

Dans cette table :

- Le champ « iditem » est la clé primaire
- Le champ « name » permet de stocker le nom de l'aliment
- Le champ « picture\_path » permet de stocker le chemin de l'image
- Le champ « category » permet de stocker toutes les catégories des aliments (céréales, fruit, légume, légumineuse, féculent)

## 6.2 Fonctions PHP

### 6.2A Ficher user.php

#### *Fonction LoginUser ()*

```
function LoginUser($nickname, $pwd)
{
    $conn = db();
    // Vérification de l'email
    $query = $conn->prepare("SELECT * FROM user WHERE `nickname` = :nickname;");
    $query->bindParam(":nickname", $nickname, PDO::PARAM_STR);
    $query->execute();
    $result = $query->fetch(PDO::FETCH_ASSOC);
    if ($result !== null && $result != "" && $result !== false) {
        // Vérification du mot de passe
        if (password_verify($pwd, $result["password"])) {
            // création des variables de session une fois que l'utilisateur est connecté
            $_SESSION['idUtilisateur'] = $result["idUser"];
            $_SESSION['nickname'] = $result["nickname"];
            return true;
        }
    }
}
```

Figure 6 Fonction LoginUser()

Le paramètre **nickname** est le pseudo que le joueur a saisie sur la page de connexion (login.php), **pwd** est son mot de passe.

Cette fonction vérifie si le pseudo et mot de passe que le joueur a saisie dans les champs sur page de connexion correspondent bien avec ceux de la base de données. Si les données sont correctes on crée alors deux variables de sessions et on retourne true.

#### *Fonction selectUserbyNickname ()*

```
function selectUserbyNickname($nickname)
{
    $conn = db();
    // Vérification de l'email
    $query = $conn->prepare("SELECT * FROM user WHERE `nickname` = :nickname;");
    $query->bindParam(":nickname", $nickname, PDO::PARAM_STR);
    $query->execute();
    $result = $query->fetch(PDO::FETCH_ASSOC);
    return $result;
}
```

Figure 7 Fonction selectUserbyNickname()

Cette fonction prend en paramètre le pseudo du joueur. Elle permet de récupérer toutes les données d'un compte utilisateur qui a comme pseudo le paramètre qu'on a donné à la fonction.

### Fonction addUser()

```
function addUser($nickname, $hash_pass,$registration_date)
{
    $pdo = db();
    if (selectUserbyNickname($nickname) == null) {
        $query = $pdo->prepare("INSERT INTO `user` (`nickname`, `password`, `registration_date`)VALUES(?, ?, ?)");
        $query->execute([$nickname, password_hash($hash_pass, PASSWORD_DEFAULT),$registration_date]);
    } else {
        return "Pseudo déjà utilisé";
    }
}
```

Figure 8 Fonction addUser()

Cette fonction permet d'ajouter (inscrire) un nouvel utilisateur dans la base de données. Elle prend en paramètre **le pseudo, le mot de passe haché et la date d'inscription**. Grâce à la fonction **selectUserByNickname()** vue précédemment, on peut vérifier si le pseudo saisi n'est pas déjà utilisé. Si la fonction retourne quelque chose, alors on renvoie un message d'erreur à l'utilisateur pour lui dire que le pseudo qu'il a saisi est déjà utilisé.

### Fonction LogOut ()

```
function LogOut()
{
    // Supprime toutes les variables de la session
    $_SESSION = [];
    // Supprime la session
    session_destroy();
    // Redirige l'utilisateur vers la page home
    header("Location: ./index.php");
    die();
}
```

Figure 9 Fonction LogOut()

Cette fonction permet de déconnecter l'utilisateur en supprimant sa session. Il est ensuite redirigé vers la page **index.php**.

### Fonction DeleteUser ()

```
function DeleteUser($idUser){
    $sql = "DELETE from user where idUser = :idUser";
    $param = [
        'idUser' => $idUser,
    ];
    dbRun($sql, $param);
    $_SESSION = [];
    // Supprime la session
    session_destroy();
}
```

Figure 10 Fonction DeleteUser()

La fonction **deleteUser ()** permet de supprimer un compte utilisateur à l'aide de son id récupéré en paramètre. Ensuite sa session est supprimée.

## 6.2B Fichier food.php

### Fonction GetAllFood ()

```
function GetAllFood(){
    $sql = "SELECT category,name,picture_path from item";
    $param = [];
    $statement = dbRun($sql, $param);
    return $statement->fetchAll(PDO::FETCH_ASSOC);
}
```

Figure 11 Fonction GetAllFood()

Cette fonction permet de récupérer la catégorie, nom et le chemin de l'image de tous les aliments contenus dans la table **item**.

### Fonction CreateGame()

```
function CreateGame($date_start,$score,$slice_count,$duration,$idUser,$status,$category,$parentIdUser){
    $sql = "INSERT INTO `game`
    (`date_start`, `date_last_update`, `score`, `slice_count`, `duration`, `idUser`, `status`, `category`, `parentIdUser`)
    value
    (:date_start,:date_last_update,:score,:slice_count,:duration,:idUser,:status,:category,:parentIdUser)";
    $param = [
        'date_start' => $date_start,
        'date_last_update' => $date_start,
        'score' => $score,
        'slice_count' => $slice_count,
        'duration' => $duration,
        'idUser' => $idUser,
        'status' => $status,
        'category' => $category,
        'parentIdUser' => $parentIdUser,
    ];
    dbRun($sql, $param);
}
```

Figure 12 Fonction CreateGame()

Cette fonction permet de créer une nouvelle partie.

Elle prend en paramètre les éléments suivants :

- **date\_start** : date de début de la partie
- **score** : le score
- **slice\_count** : le nombre d'aliments sur lesquelles le joueur à cliquer à au cours de la partie
- **duration** : le temps écoulé depuis le début de la partie
- **idUser** : l'id du joueur qui a créé la partie
- **status** : La situation de la partie (fini ou pas)
- **category** : La catégorie que le joueur a choisie
- **parentIdUser** : l'id du joueur qui a créé la partie

### *Fonction UpdateGame()*

```
function UpdateGame($score,$slice_count,$duration,$idUser,$status){  
    $sql = "UPDATE `game` SET `date_last_update` = NOW(),`score`=:score,`slice_count` = :slice_count,`duration` = :duration  
    WHERE idUser = :idUser AND status = true";  
    $param = []  
    | 'score' => $score,  
    | 'slice_count' => $slice_count,  
    | 'duration' => $duration,  
    | 'status' => $status,  
    | 'idUser' => $idUser,  
];  
dbRun($sql, $param);  
}
```

Figure 13 Fonction UpdateGame()

Cette fonction permet de mettre à jour les informations d'une partie déjà en cours.

Ces éléments sont (**score**, **slice\_count**, **duration**, **status**). La partie déjà en cours est récupérée grâce à l'id du joueur.

### *La fonction GetOnGoingGames ()*

```
function GetOnGoingGames($idUser){  
  
    $sql = "SELECT game.idGame,game.category,game.date_start,game.date_last_update,game.score,game.slice_count  
    game.duration,game.parentIdUser,user.nickname FROM game  
    INNER JOIN user ON game.idUser = user.idUser  
    where game.status = true and game.parentIdUser != :parentIdUser and game.idUser != :idUser";  
    $param = [  
        | 'parentIdUser' => $idUser,  
        | 'idUser' => $idUser,  
    ];  
    $statement = dbRun($sql, $param);  
    return $statement->fetchAll(PDO::FETCH_ASSOC);  
}
```

Figure 14 Fonction GetOnGoingGames()

Cette fonction permet de récupérer toutes les parties déjà en cours auxquelles l'utilisateur connecté ne participe pas.

### *La fonction isUserInGame ()*

```
function isUserInGame($idUser){  
    $sql = "SELECT COUNT(idUser) from game where idUser = :idUser and status = true and parentIdUser=:parentIdUser";  
    $param = [  
        | 'idUser' => $idUser,  
        | 'parentIdUser' => $idUser,  
    ];  
    $statement = dbRun($sql, $param)->fetch(PDO::FETCH_ASSOC);  
    return $statement["COUNT(idUser)"];  
}
```

Figure 15 Fonction isUserInGame()

Cette fonction permet d'empêcher un joueur de s'inscrire plusieurs fois dans une même partie.

## *La fonction nbPlayersInGame ()*

```
function nbPlayersInGame($parentIdUser,$date_start){  
    $sql = "SELECT COUNT(parentIdUser) from game where parentIdUser = :parentIdUser and status = true and date_start = :date_start";  
    $param = [  
        'parentIdUser' => $parentIdUser,  
        'date_start' => $date_start,  
    ];  
    $statement = dbRun($sql, $param)->fetch(PDO::FETCH_ASSOC);  
    return $statement["COUNT(parentIdUser)"];  
}
```

Figure 16 Fonction nbPlayersInGame

Cette fonction permet de récupérer le nombre de joueurs présent dans un partie.

Cette fonction sert à empêcher qu'il n'y ait pas plus de deux joueurs présents dans une même partie.

## *La fonction JoinGame()*

```
function JoinGame($date_start,$date_last_update,$score,$slice_count,$duration,$idUser,$status,$category,$parentIdUser){  
    if(nbPlayersInGame($parentIdUser,$date_start)!=2){  
        if(isUserInGame($idUser) < 2){  
            $sql = "INSERT INTO `game` (`date_start`, `date_last_update`, `score`, `slice_count`, `duration`, `idUser`, `status`, `category`, `parentIdUser`)  
value (:date_start,:date_last_update,:score,:slice_count,:duration,:idUser,:status,:category,:parentIdUser)";  
            $param = [  
                'date_start' => $date_start,  
                'date_last_update' => $date_last_update,  
                'score' => $score,  
                'slice_count' => $slice_count,  
                'duration' => $duration,  
                'idUser' => $idUser,  
                'status' => $status,  
                'category' => $category,  
                'parentIdUser' => $parentIdUser,  
            ];  
            dbRun($sql, $param);  
        }  
        else {  
            return "Vous avez déjà rejoint cette partie!";  
        }  
    }  
    else{  
        return "La partie est pleine!";  
    }  
}
```

Figure 17 Fonction JoinGame()

Cette fonction permet à un utilisateur de rejoindre une partie déjà en cours depuis la page **games.php**. La fonction va d'abord vérifier si le nombres de joueurs présents dans la partie ne dépasse pas 2 à l'aide de la fonction **nbPlayersInGame ()**. Si elle dépasse, la fonction retourne un message d'erreur. Ensuite elle va vérifier à l'aide de la fonction **isUserInGame ()** si un joueur n'est pas déjà présent dans la partie. S'il est déjà présent la fonction va retourner un message d'erreur encore une fois. Après toutes ces vérifications la partie du joueur voulant rejoindre la partie est enfin créée.

### *La fonction GetAllUserInformations ()*

```
function GetAllUserInformations($parentIdUser){  
    $sql = "SELECT game.score,game.idUser,game.slice_count,game.date_start,user.nickname FROM game  
    INNER JOIN user ON game.idUser = user.idUser WHERE game.parentIdUser = :parentIdUser AND game.status = true ORDER BY game.score DESC";  
    $param = [  
        ':parentIdUser' => $parentIdUser,  
    ];  
    $statement = dbRun($sql, $param)->fetchAll(PDO::FETCH_ASSOC);  
    return $statement;  
}
```

Figure 18 Fonction GetAllUserInformation()

Cette fonction permet de récupérer toutes les informations des joueurs présent dans une partie en cours. Ces informations sont :

- Le score des joueurs,
- Leurs pseudos,
- Le nombres d'aliments qu'ils ont détruits
- La date de début de la partie

### 6.2C Ficher conf.php

```
define( "DB_HOST",      "localhost");  
define( "DB_NAME",      "db_ninja");  
define( "DB_USER",      "slim_ninja_user");  
define( "DB_PASSWORD",  "TPI2023");
```

Figure 19 Fichier conf.php

Dans ce fichier on définit les constantes qui seront nécessaires pour réaliser la connexion à base de données.

## 6.2D Fichier myDB.php

```
require_once("conf.php");
/**
 * fonction qui se connecte à la base de données
 *
 * @param string sql
 * @param string param
 * @return object
 */
function db()
{
    try {
        static $myDb = null;
        if ($myDb == null) {
            $myDb = new PDO(
                "mysql:host=" . DB_HOST . ";dbname=" . DB_NAME . ";charset=utf8",
                DB_USER,
                DB_PASSWORD
            );
            $myDb->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            $myDb->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
        }
        return $myDb;
    } catch (Exception $e) {
        die("services momentanément indisponible. Merci de réessayer plus tard");
    }
}

/**
 * fonction qui exécute une requête sql
 *
 * @param string sql
 * @param string param
 * @return object
 */
function dbRun($sql, $param = null)
{
    $statement = db()->prepare($sql);

    $statement->execute($param);

    return $statement;
}
```

Figure 20 myDB.php

Dans ce fichier on va tout d'abord récupérer les données du fichier **conf.php**. Puis à l'aide de la fonction **db()** se connecter à la base de données en créant une instance d'objet PDO. Si la fonction rencontre une erreur lors de la connexion à la DB alors dans ce cas-là on affiche un message d'erreur La fonction **dbRun()** permet d'exécuter une requête SQL passée en paramètre.

## 6.3 API

Durant mon projet TPI j'ai dû réaliser de multiples api (**Application Program Interface**). L'Api va servir à échanger les données entre le site web et la base de données. Voici un schéma qui explique le fonctionnement d'une API :



Figure 21 Fonctionnement de l'API

### 6.3A Fichier getFoods.php

```
<?php
require_once('../model/functions/food.php');

switch ($_SERVER['REQUEST_METHOD']) {
    case 'GET':
        methodGET();
        break;
}
/** 
 * fonction qui encode en json tous les aliments
 *
 * @return json
 */
function methodGET()
{
    $foods = GetAllFood();
    echo json_encode([
        'foods' => $foods,
    ], JSON_UNESCAPED_UNICODE);
}
```

Figure 22 API getFoods.php

Cette fonction va permettre de récupérer tous les aliments de la base de données à l'aide de la fonction php **GetAllFood()**. Les données sont ensuite encodées en JSON.

### 6.3B Fichier getUsers.php

```
<?php
require_once('../model/functions/food.php');

switch ($_SERVER['REQUEST_METHOD']) {
    case 'GET':
        UpdateUserInformations();
        break;
}

/** 
 * fonction qui encode en json toutes les données des utilisateurs présent dans la partie
 *
 * @return json
 */
function UpdateUserInformations(){
    if($id = filter_input(INPUT_GET, 'parentUserId', FILTER_VALIDATE_INT)){
        $users = GetAllUserInformations($id);
        echo json_encode([
            'users' => $users
        ],JSON_UNESCAPED_UNICODE);
    }
}
```

Figure 23 API getUsers.php

Cette fonction permet de récupérer les données de tous les joueurs d'une partie en cours à l'aide du paramètre **\$id** (c'est l'id du joueur qui a créé la partie). Les données sont ensuite encodées en JSON.

## 6.4 Fonctions Javascript

### 6.4A Fichier user.js

*Fonction DeleteUserYesOrNo ()*

```
function DeleteUserYesOrNo(){
    const response = confirm("Voulez vous vraiment supprimer ce compte?");
    if (response) {
        location.href = "../pages/account.php?delete";
    }
}
```

Figure 24 Fonction DeleteUserYesOrNo()

Cette fonction permet d'afficher un message d'alerte avec deux boutons (Ok/Annuler) pour demander à l'utilisateur s'il veut vraiment supprimer son compte.

### 6.4B Fichier game.js

*Fonction GetRandomFoodInList ()*

```
function RandomFoodFromList(list) {
    //On filtre les aliments de la categorie
    let foodCategorieList = list.filter(item => item.category === categorie);
    if (foodCategorieList.length > 0 && Math.random() < 0.75) [
        // 75% de chance de tomber sur les aliments de la catégorie choisie
        return foodCategorieList[Math.floor(Math.random() * foodCategorieList.length)];
    ] else {
        return list[Math.floor(Math.random() * list.length)];
    }
}
```

Figure 25 Fonction GetRandomFoodInList()

Cette fonction permet de prendre un élément aléatoire de la liste des aliments. On filtre les aliments selon la catégorie que le joueur a choisi, puis si **Math.random()** (méthode JavaScript qui retourne un chiffre aléatoire) retourne un chiffre plus petit que 0.75, alors on retourne l'aliments de la catégorie , sinon on retourne un autre aliments.

*Fonction CreateFood()*

```
function createFood() [
    let randomFood = RandomFoodFromList(FoodList);
    let food = {
        x: Math.random() * Math.floor(Math.random() * (canvas.width - (FoodHeight) - FoodWidth)) + FoodWidth,
        y: -FoodHeight,
        width: FoodWidth,
        height: FoodHeight,
        src: "../sprites/" + randomFood.picture_path,
        speed: Speed,
        name: randomFood.name,
        category: randomFood.category,
    };
    return food;
]
```

Figure 26 Fonction CreateFood()

La fonction suivante permet de créer un aliment. Elle retourne un objet qui contient la position x de l'aliment (aléatoire), la position y qui est tout le temps la même, la largeur et la longueur de l'image qui sont proportionnelle à la taille du canvas. La source de l'image, la vitesse, le nom et la catégorie sont récupérée depuis la fonction **RandomFoodFromList()**.

### Fonction draw()

```
function draw() {
    // Effacer le canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Dessine chaque aliment
    for (let i = 0; i < foods.length; i++) {
        let food = foods[i];
        let img = new Image();
        img.src = food.src;
        ctx.drawImage(img, food.x, food.y, food.width, food.height);

        // Afficher le nom de l'aliment
        ctx.font = "bold 13px Comic Sans MS";
        ctx.fillText(food.name, food.x + food.width/3, food.y + food.height*1.2);

        // Fait tomber l'aliment avec la vitesse
        food.y += food.speed;

        // Si l'aliment atteint le bas de l'écran, on le supprime de la liste
        if (food.y > canvas.height - food.height) {
            if (food.category == categorie) {
                Score -= 2;
                if (Score < 0) {
                    Score = 0;
                    GameOver();
                }
            }
            sendDataToPhp(categorie, Score, Slice_count, 1);
            foods.splice(i, 1);
            break;
        }
    }
    // On vérifie si le nombre des aliments est plus petit que le nombre max des aliments
    if (foods.length < nbFood) {
        foods.push(createFood());
    }
}
```

Figure 27 Fonction draw()

Cette fonction permet de dessiner un aliment ainsi que son nom sur le canvas. On ajoute la vitesse à l'aliment. Ensuite on va aussi vérifier si un des aliments de la liste **foods** atteint le bas de l'écran. Si l'aliment qui atteint le bas de l'écran fait partie de la catégorie que le joueur a choisie, le joueur perd deux points et l'aliment est supprimé de la liste. Si son score est négatif alors on affiche l'écran de la fin du jeu. Puis finalement on va vérifier si le nombre d'aliments qui tombent sur l'écran est plus petit que le nombre maximum d'aliments qui

peuvent tomber simultanément. Si non on ajoute un nouvel aliment dans la liste contenant tous les aliments(**foods**). La fonction est appelée toutes les 10 millisecondes.

```
let drawInterval = setInterval(draw, 10);
```

Figure 28 setInterval()

*Fonction isPointerInsideFood ()*

```
function isPointerInsideFood(x, y, rect) {
    return x >= rect.x && x <= rect.x + rect.width &&
           y >= rect.y && y <= rect.y + rect.height;
}
```

Figure 29 isPointerInsideFood

Cette fonction permet de vérifier si le curseur est à l'intérieur d'un des aliments présents sur l'écran. Elle prend en paramètre la position verticale de la souris, la position horizontale de la souris et l'aliment.

### Fonction handleClickEvent()

```
function handleClickEvent(event) {
    // Les coordonnées exactes du curseur
    let clicX = event.offsetX;
    let clicY = event.offsetY;

    // Vérifier si le cursor est à l'intérieur d'un aliment
    for (let i = 0; i < foods.length; i++) {
        let food = foods[i];
        if (isPointerInsideFood(clicX, clicY, food)) {
            // Supprimer l'aliment et on met à jour les informations
            if (food.category == categorie) {
                Speed += 0.1;
                Score++;
                nbFood = Math.floor(1 + (0.2 * (Speed - 2) / 0.1));
                SpeedElement.innerText = "Vitesse: *" + Math.round(Speed * 10) / 10;
                Slice_count++;
                ShowUserInformations();
                sendDataToPhp(categorie, Score, Slice_count, time, 1);

            } else {
                if (food.category == "malbouffe") {
                    Score = 0;
                    GameOver();
                }
                Score -= 2;
                Slice_count++;
                ShowUserInformations();
                if (Score < 0) {
                    Score = 0;
                    GameOver();
                }
            }
            sendDataToPhp(categorie, Score, Slice_count, time, 1);
            //Supprimer l'aliment du tableau
            foods.splice(i, 1);
            break;
        }
    }
}
```

Figure 30 handleClickEvent()

```
canvas.addEventListener('click', handleClickEvent, false);
```

Figure 31 Évènement click

Cette fonction se produit à chaque fois que le joueur clique sur le canvas.

La fonction suivante permet de vérifier si l'utilisateur a appuyé sur un des aliments de la liste. S'il a appuyé sur le bon aliment de la catégorie, le score du joueur augmente ainsi que sa vitesse et le nombre d'aliments qu'il a découpé. Le nombre maximum d'aliments qui peuvent tomber en même temps augmente selon la vitesse. En revanche s'il sélectionne le mauvais aliment il perd deux points. Si son score est négatif il perd la partie. Cependant si le joueur appuie sur un des aliments faisant partie de la catégorie « **malbouffe** », il perd

directement la partie. Ces données sont ensuite envoyées en PHP à l'aide de la fonction **sendDataToPhp ()** pour être ensuite envoyé à la base de données.

#### Fonction addAndShowUsersSpeed()

```
function AddAndShowUsersSpeed() {
    UpdateUsers();
    let userScore = JSON.parse(sessionStorage.getItem("ListUsers"));
    if (userScore.length > 1) {
        isMultiplayer = true;
        // Si le joueur a le score le plus petit alors la vitesse augmente plus vite
        if (userScore[0]["idUser"] == idUser) {
            speed += 0.1;
            SpeedElement.innerText = "Vitesse: *" + Math.round(speed * 10) / 10;
        }
    }
}
```

Figure 32 Fonction addAndShowUsersSpeed()

Cette fonction permet de récupérer toutes les données des utilisateurs présent dans la partie à l'aide de la fonction **UpdateUsers()** et à les afficher. Si le nombre des utilisateurs présent dans la partie est plus grand que 1 alors on augmente la vitesse du joueur qui a le plus petit score.

#### Fonction reziseCanvas ()

```
function resizeCanvas() {
    canvas.width = window.innerWidth * 0.8;
    canvas.height = window.innerHeight * 0.8;
}
```

Figure 33 Fonction resizeCanvas()

```
window.addEventListener('resize', resizeCanvas, false);
```

Figure 34 Évènement resize

Cette fonction se produit lorsque le joueur essaie de redimensionner la taille de l'écran.

Elle permet de changer la taille de l'écran est en même temps à le rendre responsive.

### Fonction GameOver()

```
function GameOver() {
    let userScore = JSON.parse(sessionStorage.getItem("ListUsers"));
    // Supprime la variable de l'url
    url.searchParams.delete("category");
    history.replaceState(null, "", url);
    sendDataToPhp(categorie, Score, Slice_count, 0);
    clearInterval(UpdateInformations);
    clearInterval(drawInterval);
    canvas.remove();
    if (userScore.length < 2) {
        if(isMultiplayer){
            document.getElementById("winner").innerText = "Vous avez gagné la partie!";
        }
    }
    document.getElementById("duration").innerText = document.getElementById("time").textContent;
    GameOverMenu.style.visibility = "visible";
    Informations.style.visibility = "hidden";
    GameOverMenuButton.addEventListener('click', function (event) {
        location.href = "./categories.php";
    });
}
```

Figure 35 Fonction GameOver()

Cette fonction permet d'afficher l'écran de fin de jeu lorsque le joueur perd la partie, elle permet dans un second temps d'afficher le gagnant de la partie. On affiche aussi la durée de la partie et un bouton retour qui le redirige vers la page **cateogries.php**

```
window.addEventListener('beforeunload', function (event) {
    sendDataToPhp(categorie, Score, Slice_count, time, 0);
});
```

Figure 36 Évènement beforeunload

Si le joueur quitte ou recharge la page on met que la partie est terminé.

## 6.4C Fichier UpdateUsers.js

```
export function UpdateUsers() {
  const config = {
    method: "get",
  };
  var url = new URL(document.location.href);
  var search_params = new URLSearchParams(url.search);
  //On vérifie si le paramètre id existe dans l'url
  if (search_params.has('parentUserId')) {
    var id = search_params.get('parentUserId');
    // On récupère le résultat que retourne l'api getUsers
    fetch('../api/getUsers.php?parentUserId=' + id, config)
      .then(response => { return response.json() })
      .then(json => [
        let nicknames = "";
        let time = "";
        json.users.forEach(element => {
          nicknames += "Joueur: " + element["nickname"] + " Score: " + element["score"] + "\n";
          time = element["date_start"];
        });
        // On affiche les données des utilisateurs
        document.getElementById("nickaname").innerText = nicknames;
        let now = new Date();
        let date_start = new Date(time);

        // Calculer la différence en millisecondes entre les deux dates
        let difference = now.getTime() - date_start.getTime();

        // Convertir la différence en millisecondes en secondes, minutes, heures et jours
        let seconds = Math.floor(difference / 1000);
        let minutes = Math.floor(seconds / 60);
        let hours = Math.floor(minutes / 60);

        // Calculer le temps restant dans chaque unité
        let remaningSeconds = seconds % 60;
        let remaningMinutes = minutes % 60;
        let remaningHours = hours % 24;

        // On affiche le temps restant
        document.getElementById("time").innerText = "Temps : " + remaningHours + ":" + remaningMinutes + ":" + remaningSeconds;
        sessionStorage.setItem('ListUsers', JSON.stringify(json.users));
      ]);
  }
}
```

Figure 37 Fonction UpdateUsers()

Cette fonction permet de récupérer toutes les données des utilisateurs présent dans une partie depuis le fichier **getUsers.php**. Cette fonction va aussi permettre d'afficher les pseudos, les scores des joueurs et le temps écoulé qui est calculer de manière suivante (temps actuel – le temps de début de la partie). Le temps est obtenu est millisecondes, on va le convertir en secondes, minutes et heures puis l'afficher. Cette fonction est finalement exportée vers le fichier **game.js**.

## 6.4D Fichier addBadFood.js

### Fonction addBadFood ()

```
export function addBadFood(list) [
  list.push({ category: "malbouffe", name: "Fausse-tomate", picture_path: "Fausse-tomate.png" });
  list.push({ category: "malbouffe", name: "Fausse-aubergine", picture_path: "Fausse-aubergine.png" });
]
```

Figure 38 Fonction addBadFood()

Cette fonction prend en paramètre la liste des aliments. Elle permet d'ajouter deux aliments malsains. Elle est exportée vers le fichier **game.js**.

## 6.4E Fichier getFoods.js

*Fonction GetAllFood()*

```
function GetAllFood() {
    const config = {
        method: "get",
    };

    fetch("./api/getFoods.php", config)
        .then(response => { return response.json() })
        .then(json => {
            localStorage.setItem("Foods", JSON.stringify(json.foods));
        });
}
```

Figure 39 Fonction GetAllFood()

Cette fonction permet de récupérer tous les aliments de la base de données à l'aide d'une api **getFoods.php**. Ensuite on va stocker la liste des aliments encodé en JSON dans un localstorage () pour pouvoir la récupérer dans le fichier **game.js**.

## 6.4F Fichier sendDataToPhp.js

*Fonction SendDataToPhp*

```
export function sendDataToPhp(categorie, score, slice_count, time, status) {
    let dataToSend = JSON.stringify([score, slice_count, time, status]);
    fetch('../pages/game.php?category=' + categorie, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            'X-Requested-With': 'XMLHttpRequest'
        },
        body: 'data=' + encodeURIComponent(dataToSend)
    })
        .then(response => response.text())
        .then(result => {
            document.getElementById("result").innerHTML = result;
        })
        // S'il y a une erreur on l'affiche dans la console
        .catch(error => {
            console.error('Erreur:', error);
        });
}
```

Figure 40 Fonction SendDataToPhp()

Cette fonction va envoyer des données de la partie en cours depuis JavaScript jusqu'à la page **game.php** via une requête POST. Ces données sont envoyées dans un tableau qui est encodé en JSON. La fonction est exportée vers la page **game.js**.

```
if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) && strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest') {  
    if (isset($_POST['data'])) {  
        $receivedData = json_decode($_POST['data']);  
        UpdateGame($receivedData[0], $receivedData[1], $receivedData[2], $_SESSION['idUtilisateur'], $receivedData[3]);  
    } else {  
        echo "Aucune donnée reçue.";  
    }  
    exit;  
}
```

Figure 41 Vérification de la réception de la requête HTTP

Dans la page **game.php** on va vérifier si on a bien reçu les données. Si oui on va mettre à jour les données de la partie dans la base de données à l'aide de la fonction **UpdateGame()**

## 7. Problèmes rencontrés

Lorsque j'ai commencé à travailler sur mon projet TPI, j'ai rapidement constaté qu'il y avait un parti multijoueur dans mon cahier de charges. Au début je ne comprenais pas bien cette notion et je m'étais essentiellement concentré sur la conception des fonctionnalités principales du jeu. J'ai bien pu avancer jusqu'à ce que je me heurte à la partie multijoueur. J'ai rapidement pris contact avec mon maître de TPI pour que l'on se fixe un rendez-vous. Nous nous sommes rencontrés le jour même, après une brève discussion j'ai finalement compris ce que l'on attendait de moi. À part ce petit point bloquant, je n'ai pas rencontré d'autres problèmes au cours du développement de ce projet.

## 8. Maquettes

Au cours de mon TPI, j'ai dû créer des maquettes afin d'avoir une vision globale de ce que je devais accomplir.

## 8.1 Page Inscription

**Inscription:**

**Pseudo :**

**Mot de passe :**

[J'ai déjà un compte!](#)

Figure 42 Page Inscription

## 8.2 Page Login

**Connexion**

**Pseudo :**

**Mot de passe :**

[Je n'ai pas de compte!](#)

Figure 43 Page Login

## 8.3 Page Index

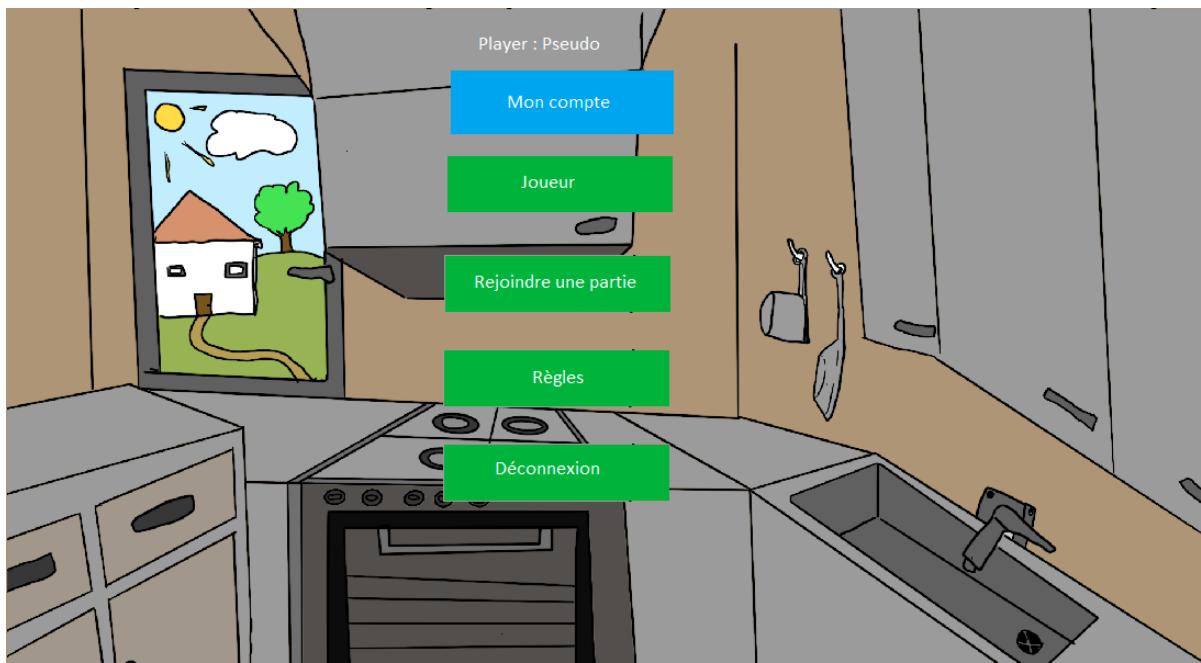


Figure 44 Page Login

## 8.4 Page Account

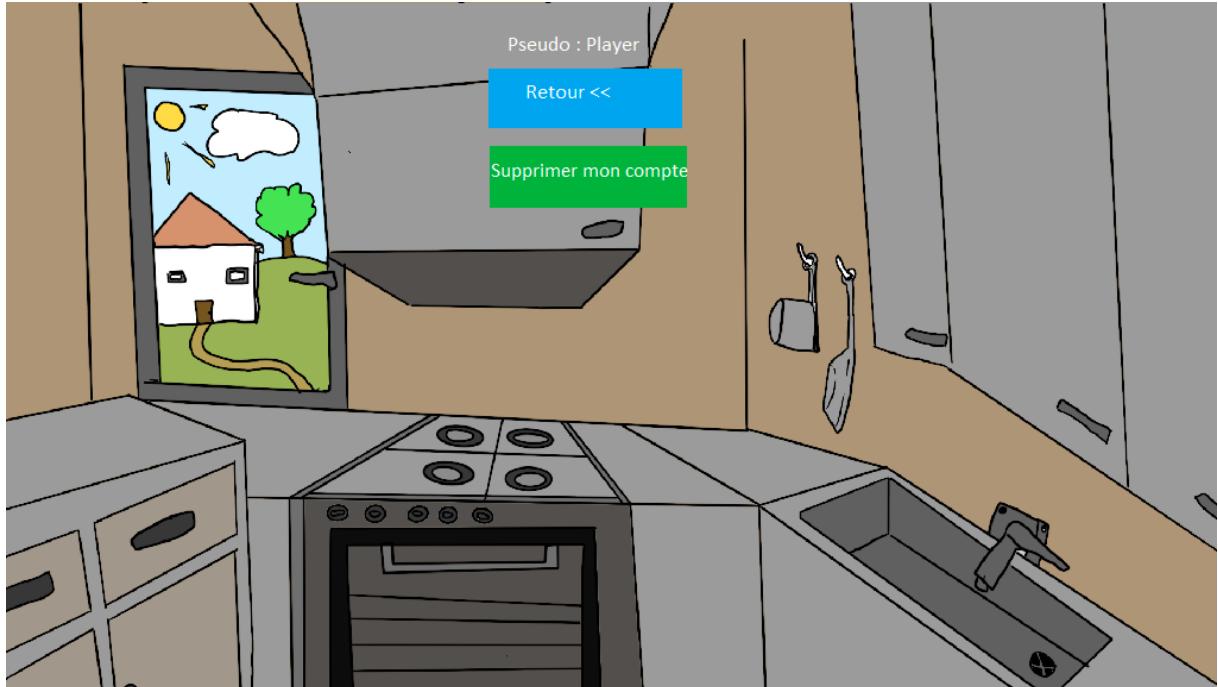


Figure 45 Page Account

## 8.5 Page Rules

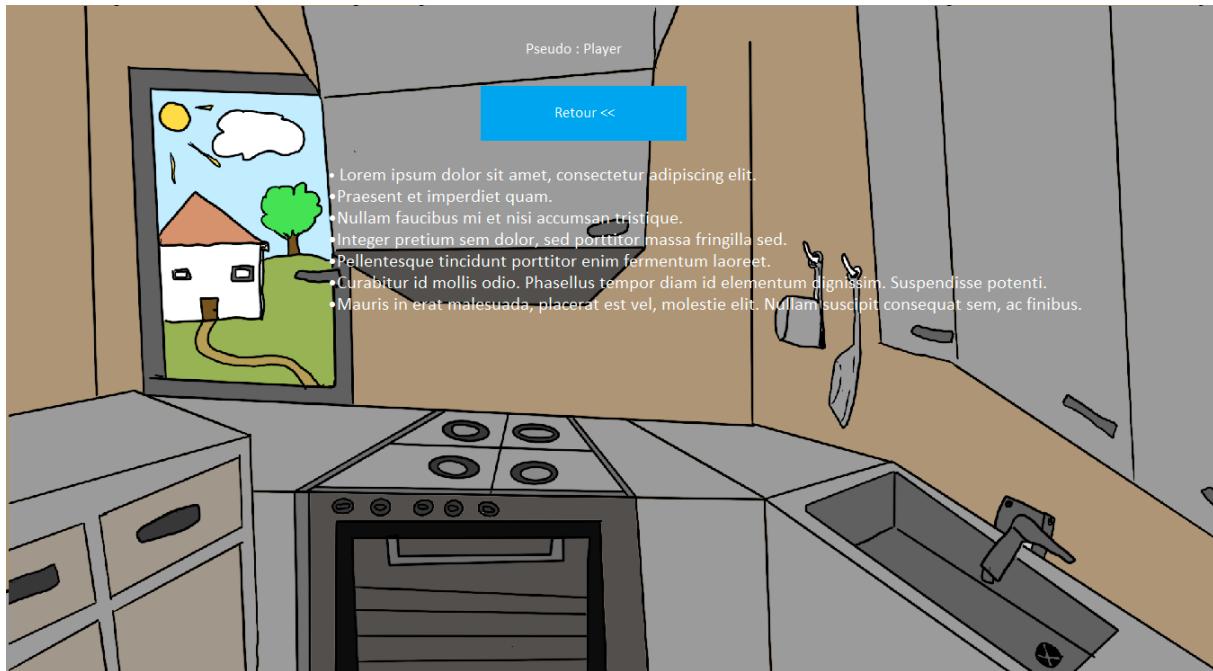


Figure 46 Page Rules

## 8.6 Page Games

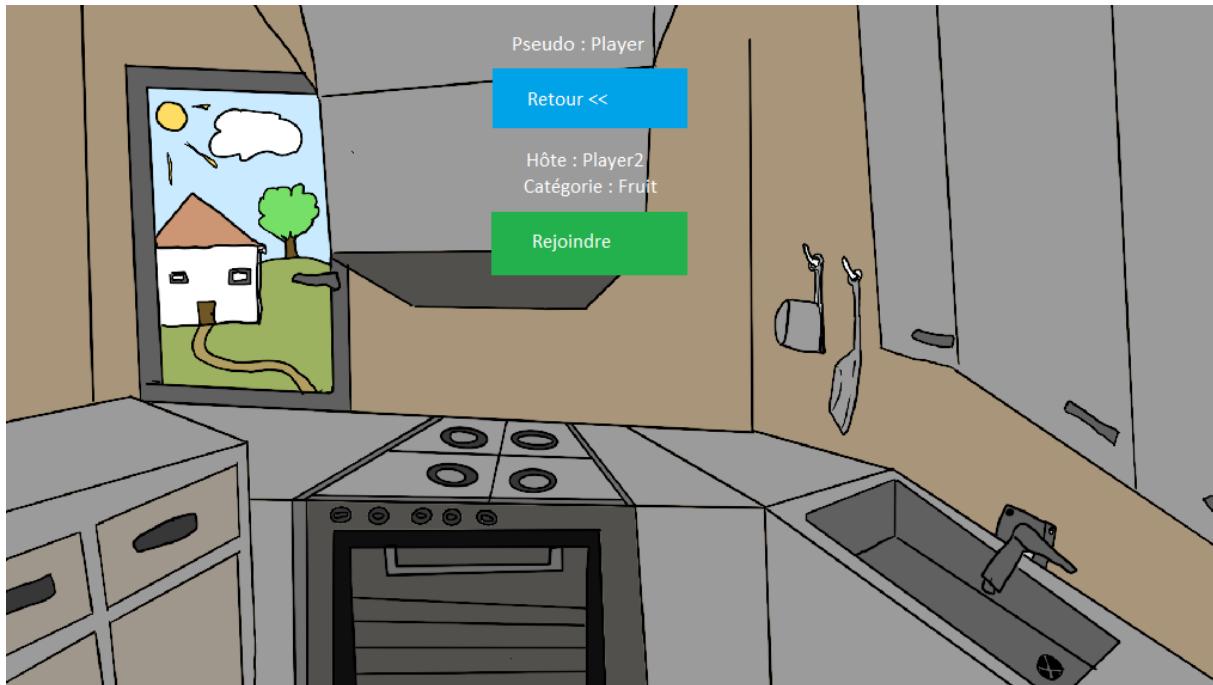


Figure 47 Page Games

## 8.7 Page Game



Figure 48 Page Game

## 8.8 Fin d'une partie

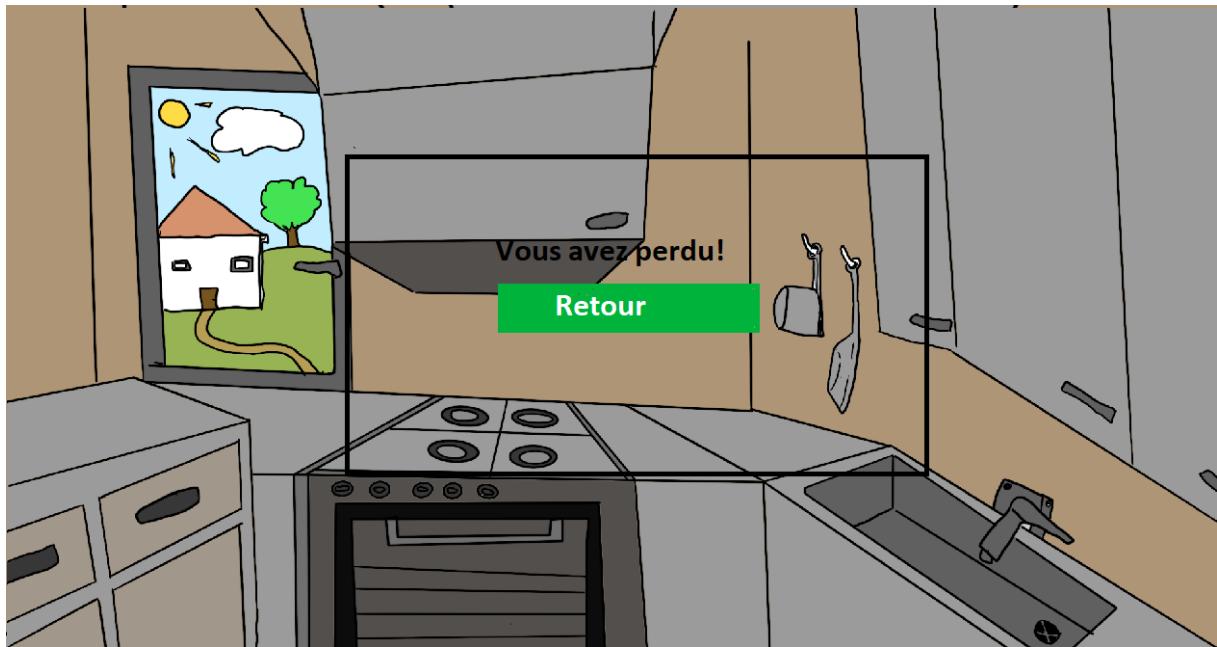


Figure 49 La fin du jeu

## 8.9 Schéma des groupes d'aliments

Fruit	
Légume	
Légumineuse	
Céréale	
Féculent	
Malbouffe	

Figure 50 Schéma des catégories d'aliments

## 8.10 Schéma de navigation du site

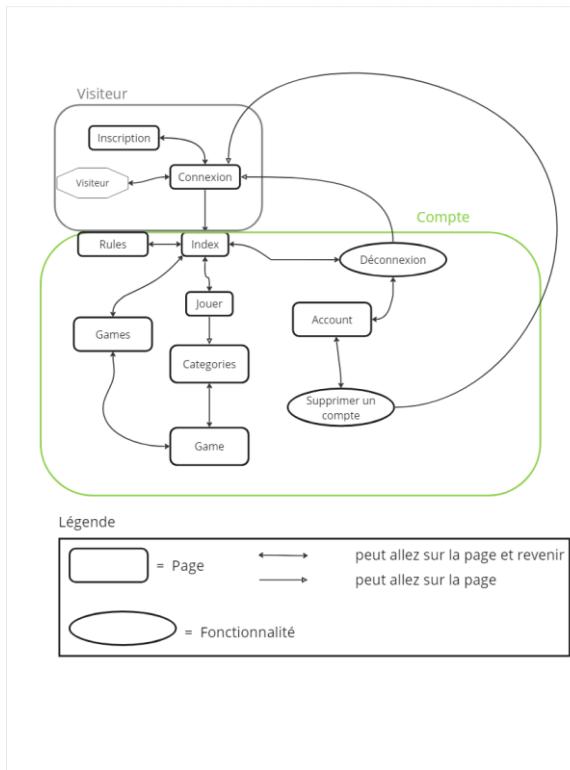


Figure 51 Schéma de navigation du site

# 9. Plan de tests et rapport de tests

## 9.1 Configuration :

### -matériel :

- Nom du model : EliteDesk 800 G4
- Mémoire RAM : 32 Go
- Processeur : Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz
- Carte graphique : Intel HD Graphics 630
- Windows 10 Éducation (64-Bit)
- Tous les tests sont réalisés sur un écran avec une résolution de 2560x1440

### -logiciel

- Serveur web : Wamp Server
- Système d'exploitation : Windows 10 Éducation
- MySQL : 5.7.36
- PHP : 7.4.26

## 9.2 Tests :

### 9.2A Fonctions PHP

#### 9.2A1 : Gestion de l'utilisateur

Nom test	Descriptif	Version test	Date
Création d'un compte utilisateur	Le but est de tester la fonction addUser () et vérifier si elle marche bien	1.0.0	02.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Allez sur la page de l'inscription	Appuyez sur le lien « Je n'ai pas de compte ? »	L'utilisateur est bien redirigé vers la page	Ok	
2	Tester la création de compte	Remplir les champs Pseudo : Player Mot de passe : Super Appuyer sur le bouton valider	L'utilisateur est redirigé vers la page de connexion est un message de validation lui est affiché	Ok	
3	Tester si le message d'erreur s'affiche	Remplir les champs avec les mêmes données que dans la	Le message d'erreur s'affiche bien	Ok	

		deuxième étape. Appuyer sur le bouton valider			
--	--	--	--	--	--

Nom test	Descriptif	Version test	Date
Connexion à un compte utilisateur	Le but est de tester la fonction LoginUser () et vérifier si elle marche bien	1.0.0	02.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la connexion fonctionne	Remplir les champs Pseudo : Player Mot de passe : Super	L'utilisateur est bien redirigé vers la page index.php	Ok	
2	Tester la création de compte	Remplir les champs Pseudo : Player Mot de passe : Super Appuyer ensuite sur le bouton valider	L'utilisateur est redirigé vers la page de connexion est un message de validation lui est affiché	Ok	
3	Tester si le message d'erreur s'affiche	Appuyer sur le bouton valider avec les champs vides	Le message d'erreur s'affiche bien	Ok	
4	Tester si le deuxième message d'erreur s'affiche	Remplir les champs avec les données suivantes Pseudo : Stefan Mot de passe : Super	Le message d'erreur s'affiche bien	Ok	

Nom test	Descriptif	Version test	Date
Suppression d'un compte utilisateur	Le but est de tester la fonction deleteUser () et vérifier si elle marche	1.0.0	02.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Allez sur la page account.php	Appuyer sur le bouton mon compte sur la page du menu principale	L'utilisateur est bien redirigé vers la page account.php	Ok	
2	Tester si la suppression de compte fonctionne	Appuyer sur le bouton supprimer mon compte puis appuyer sur le bouton OK lorsque le pop-up apparaît	L'utilisateur est redirigé vers la page de connexion et son compte est bien supprimé de la base de données	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction LogOut ()	Le but est de tester la fonction LogOut () et de vérifier si elle fonctionne bien.	1.0.0	02.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la déconnexion de l'utilisateur fonctionne	Allez sur la page index.php, puis appuyer sur le bouton déconnexion	L'utilisateur est bien redirigé vers la page login.php	Ok	

## 9.2A2 Gestion du jeu

Nom test	Descriptif	Version test	Date
Test de la fonction CreateGame()	Le but est de tester la fonction CreateGame() et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction CreateGame() fonctionne	Allez sur la page index.php, puis appuyer sur le bouton jouer, ensuite appuyer le bouton fruit	La partie est créée dans la base de donné	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction UpdateGame()	Le but est de tester la fonction UpdateGame() et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction UpdateGame() fonctionne lorsque l'utilisateur appuie sur un fruit	Allez sur la page index.php, puis appuyer sur le bouton jouer, ensuite appuyer le bouton fruit	Les donnés de la partie sont bien mises à jour dans la base de données	Ok	
2	Tester si la fonction UpdateGame () fonctionne lorsqu'un fruit touche le sol	Allez sur la page index.php, puis appuyer sur le bouton jouer, ensuite appuyer le bouton fruit	Les données de la partie sont bien mises à jour dans la base de données	Ok	
3	Tester si la fonction UpdateGame() fonctionne lorsque le joueur quitte ou recharge la page	Allez sur la page index.php, puis appuyer sur le bouton jouer, ensuite appuyer le bouton fruit	Les données de la partie sont bien mises à jour dans la base de données	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction GetOnGoingGames ()	Le but est de tester la fonction GetOnGoingGames () et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction GetOnGoingGames() fonctionne	Lancer une partie en mode fruit. Ouvrir une navigation privée et se connecter avec un autre compte. Appuyer sur le bouton Rejoindre une partie	Les parties en cours s'affichent bien	Ok	

Contenu du test :

Nom test	Descriptif	Version test	Date
Test de la fonction isUserInGame()	Le but est de tester la fonction isUserInGame () et vérifier si elle marche	1.0.0	15.05.2023

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction isUserInGame() fonctionne	Lancer une partie en mode fruit. Ouvrir une navigation privé et se connecter avec le même compte. Appuyer sur le bouton Rejoindre une partie puis cliquer sur le bouton rejoindre	L'utilisateur est renvoyé sur la page games.php. Un message d'erreur lui est affiché	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction nbPlayersInGame()	Le but est de tester la fonction nbPlayersInGame() et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction nbPlayersInGame() fonctionne	Lancer une partie en mode fruit. Ouvrir deux navigateurs en mode privé et se connecter avec deux comptes différents. Appuyer sur le bouton Rejoindre une partie puis rejoindre avec les deux comptes	Le troisième utilisateur est renvoyé sur la page games.php avec un message d'erreur	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction JoinGame ()	Le but est de tester la fonction JoinGame () et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction JoinGame () fonctionne	Lancer une partie en mode fruit avec un premier compte. Ouvrir une nouvelle fenêtre en mode privé et se connecter avec un autre compte. Appuyer sur le bouton rejoindre une partie puis cliquer sur le bouton rejoindre	La partie est créé dans la base de données. Le champ parentIdUser contient bien L'id du joueur qui a créé la partie	Ok	

Nom test	Descriptif	Version test	Date
Test de la fonction GetAllUserInformations()	Le but est de tester la fonction GetAllUserInformations() et vérifier si elle marche	1.0.0	15.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction GetAllUserInformations() fonctionne	Lancer une partie en mode fruit avec un compte.	La requête retourne bien toutes les infos de l'utilisateur	Ok	

## 9.2B Fonctions JavaScript

### Fichier *getFood.js*

Nom test	Descriptif	Version test	Date
Test de la fonction GetAllFood()	Le but est de tester la fonction GetAllFood() et vérifier si elle marche	1.0.0	03.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction GetAllFood () fonctionne	Allez sur la page index.php, puis appuyer sur le bouton jouer, ensuite appuyer le bouton fruit	La liste des aliments apparait bien dans la console du javascript	NOK	La liste des aliments ne s'affichent pas
2	Tester si les aliments s'affichent bien dans la console	Tester l'étape 1	La liste des aliments apparait bien dans la console du javascript	Ok	

### *Fichier addBadFood.js*

Nom test	Descriptif	Version test	Date
Test de la fonction addBadFood ()	Le but est de tester la fonction addBadFood() et vérifier si elle marche	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction addBadFood () fonctionne	Allez sur index.php puis appuyer sur le bouton jouer puis appuyer sur le bouton Fruit	Les deux aliments sont ajoutés dans la liste FoodList	Ok	

### *Fichier user.js*

Nom test	Descriptif	Version test	Date
Test de la fonction DeleteUserYesOrNo()	Le but est de tester la fonction DeleteUserYesOrNo () et vérifier si elle marche	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction DeleteUserYesOrNo () fonctionne	Allez sur index.php puis appuyer sur le bouton mon compte. Appuyer ensuite sur le bouton Supprimer mon compte	Un pop-up est affiché	Ok	

### *Fichier sendDataToPhp.js*

Nom test	Descriptif	Version test	Date
Test de la fonction sendDataToPhp()	Le but est de tester la fonction sendDataToPhp () et vérifier si elle marche	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction sendDataToPhp () fonctionne Lorsque le joueur appuie sur un fruit	Lancer un jeu en mode fruit, puis appuyer sur un fruit	Les données de la partie sont mis à jour	Ok	
2	Tester si la fonction sendDataToPhp () fonctionne lorsqu'un fruit touche le sol	Lancer un jeu en mode fruit, puis Attendre qu'un fruit touche le sol	Les données de la partie sont mises à jour	Ok	
3	Tester si la fonction sendDataToPhp () fonctionne lorsqu'on recharge la page	Lancer un jeu en mode fruit, puis appuyer sur un fruit	Les données de la partie sont mises à jour	Ok	

### *Fichier updateUsers.js*

Nom test	Descriptif	Version test	Date
Test de la fonction updateUsers()	Le but est de tester la fonction updateUsers () et vérifier si elle marche	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la fonction updateUsers() fonctionne Lorsque le joueur appuie sur un fruit	Lancer un jeu en mode fruit, puis appuyer sur un fruit	Le score, le pseudo , le temps écoulé sont bien affiché	Ok	
2	Tester si la fonction updateUsers() fonctionne Lorsqu'un autre joueur rejoint la partie	Lancer un jeu en mode fruit, puis appuyer sur un fruit. Faire la même chose mais avec un autre compte dans une navigation privé	Les score, les pseudo des joueurs s'affichent en permanence	Ok	

### Fichier game.js

Nom test	Descriptif	Version test	Date
Test de la création des aliments	Le but est de tester si les aliments apparaissent sur l'écran de manière aléatoire	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si les fruits sont créés lorsque le joueur lance une partie	Lancer un jeu en mode fruit	Les fruits apparaissent de manière aléatoire sur l'écran	Ok	

Nom test	Descriptif	Version test	Date
Test de déplacement d'aliments	Le but est de tester si les aliments tombent du haut vers le bas	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si les aliments tombent du haut vers le bas	Lancer un jeu en mode fruit	Les fruits tombent du haut vers le bas	Ok	
2	Tester si les aliments sont supprimés lorsqu'il touche le bas de l'écran	Lancer un jeu en mode fruit, attendre qu'un fruit touche le bas de l'écran	Le/s fruits sont supprimé	Ok	

Nom test	Descriptif	Version test	Date
Test de changement du score	Le but est de tester si le score s'incrémente ou diminue	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si score s'incrémente lorsque le joueur appuie sur le bon aliment de la catégorie	Lancer un jeu en mode fruit, appuyer sur une pomme ou une poire	Le score est augmenté de 1	Ok	
2	Tester si le score diminue lorsque le joueur appuie sur un aliment qui ne fait pas partie de la catégorie	Lancer un jeu en mode fruit, Appuyer sur un autre aliment que pomme ou poire	Le score est diminué de -2	Ok	

Nom test	Descriptif	Version test	Date
Test de la fin du jeu	Le but est de tester si le menu de fin de jeu s'affiche	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si le menu de fin de jeu s'affiche	Lancer un jeu en mode fruit,	Le menu de fin de jeu est affiché	Ok	

	lorsque le score du joueur devient négatif	attendre qu'une pomme ou poire touche le bas de l'écran ou appuyer sur un aliment d'une autre catégorie			
2	Tester si le menu de fin de jeu s'affiche lorsque le joueur appui sur un aliment malsain	Lancer un jeu en mode fruit, Jouer le jeu jusqu'à ce qu'un aliment malsain apparait (fausse-tomate, fausse-aubergine). Appuyer dessus ensuite.	Le menu de fin de jeu est affiché	Ok	

Nom test	Descriptif	Version test	Date
Test de l'augmentation de nombres de fruits	Le but est de tester le nombre d'aliments augmente	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si le nombres d'aliments augmente par rapport à la vitesse	Lancer un jeu en mode fruit, Jouer le jeu	Plus la vitesse du joueur augmente, plus il y a aura d'aliments	Ok	

Nom test	Descriptif	Version test	Date
Test de l'augmentation la vitesse du joueur ayant le score le plus petit	Le but est de tester si la vitesse du joueur qui a le score le plus petit augmente plus que la vitesse du l'autre joueur	1.0.0	16.05.2023

Contenu du test :

Numéro de l'Étape	Nom Étape	Action à réalisé	Résultat attendu	Status	Observation
1	Tester si la vitesse du joueur qui a le score le plus petit augmente plus que la vitesse de l'autre joueur	Lancer un jeu en mode fruit avec deux comptes différents dans deux fenêtres différentes (navigation privé).	La vitesse du joueur qui a le score le plus petit augmente plus vite	Ok	

## 10. Conclusion

### 10.1 Futurs améliorations

Description
Ajouter une véritable connexion multijoueur permettant à chacun des deux joueurs de voir en temps réel l'écran de l'autre joueur.
Augmenter le nombre de joueurs qui peuvent rejoindre une partie.
Ajouter plus de catégories et d'aliments.
Ajouter une animation après qu'on a cliqué sur l'aliment.
Lorsqu'un joueur quitte une partie il ne peut plus la rejoindre
Faire en sorte que les aliments continuent de tomber même lorsqu'on est plus sur la page.
Des bonus qui pourraient par exemple baisser la vitesse de défilement du joueur.
Des malus qui pourraient par exemple augmenter la vitesse de défilement du joueur.

### 10.2 Bilan personnel

Ce projet TPI m'a permis de réaliser un projet dans son intégralité. Malgré quelques moments où ma motivation a faibli, j'ai réussi à rester motivé jusqu'à la fin du projet.

Au début de ce projet je n'avais pas très bien compris ce que l'on attendait de moi surtout pour la partie multijoueur. Mais grâce aux explications de Monsieur, François j'ai pu finalement comprendre le travail qui était attendu de moi.

Ensuite, j'avais mal estimé le temps que je prendrais pour réaliser la documentation technique mais j'ai pu finalement la terminé à temps.

Pour résumer mon TPI j'ai pu acquérir de nombreuses connaissances (mieux gérer le temps que j'ai à disposition). Ces connaissances me seront très bénéfique pour la suite de mes études.

Pour conclure, j'ai adoré réaliser le projet **Slim-Ninja**

## 11.Glossaire

Terme	Description
MCD	Modèle conceptuel de données : Schémas représentant les tables de la base de données et leurs relations.
MLD	Modèle logique de données : Modèle textuel de la structure de la base de données avec les clés primaires et étrangères.
BD	Base de données
SQL	(Structured Query Language). SQL est un langage de programmation qui permet de manipuler les bases de données
JSON	C'est un format d'échange de données
LocalStorage	Stockage web locale

## 12.Bibliographie

Référence Javascript

<https://www.w3schools.com>

<https://openclassrooms.com/fr/>

Référence PHP

<https://www.php.net>

<https://stackoverflow.com>

Site que j'ai utilisé pour trouver de quelle catégorie faisant partie les différents aliments

<https://www.alimentarium.org/fr/node/5507>

Site ou j'ai réalisé le schéma pour représenter les différents groupes d'aliments

<https://tiermaker.com>

Les spirites et le fond d'écran du jeu ont été fait avec le site suivant :

<https://www.pixilart.com>

Pour faire le déplacement des aliments je me suis inspiré d'un ancien projet que j'avais fait en 2<sup>ème</sup> année en JavaScript.

Pour envoyer les données de javascript en PHP j'ai utilisé le code que j'ai trouvé sur ce forum :

<https://www.webrankinfo.com/forum/t/passer-une-variable-js-vers-php.194183/>

Pour faire le schéma de navigation du site et le schéma de l'API j'ai utilisé le site suivant :

<https://miro.com/app/dashboard/>

Pour fusionner le code source avec la documentation technique j'ai utilisé le site suivant :

<https://www.adobe.com/fr/acrobat/online/merge-pdf.html>

Pour faire la fonction mathématique qui fait augmenter le nombre d'aliments par rapport à la vitesse j'ai utilisé le site suivant avec l'aide de **Mr.Folly Sodoga** :

<https://www.geogebra.org/?lang=fr>

## 12.1 ChatGPT

Pour pouvoir calculer le temps écoulé depuis le début de la partie J'ai utilisé le ChatGPT. Voici la conversion que je eu avec lui :

S peux tu calculer la diffrence de temps de maintenant - 2023-05-14 11:22:14 dans le même format en javascript

Oui, voici comment calculer la différence de temps entre maintenant et "2023-05-14 11:22:14" en JavaScript :

```
javascript
// Obtenir la date et l'heure actuelles
let maintenant = new Date();

// Créer un objet Date pour "2023-05-14 11:22:14"
let dateLimite = new Date("2023-05-14T11:22:14");

// Calculer la différence en millisecondes entre les deux dates
let difference = dateLimite.getTime() - maintenant.getTime();

// Convertir la différence en secondes, minutes, heures et jours
let secondes = Math.floor(difference / 1000);
let minutes = Math.floor(secondes / 60);
let heures = Math.floor(minutes / 60);
let jours = Math.floor(heures / 24);

// Formater la différence de temps comme une chaîne de caractères
let differenceTemps = jours + " jours, " + heures % 24 + " heures, " + minutes % 60 + " minutes, " + secondes % 60 + " secondes";

console.log(differenceTemps);
```

Cela affichera la différence de temps sous forme de chaîne de caractères avec le nombre de jours, heures, minutes et secondes restants jusqu'à "2023-05-14 11:22:14".

Figure 52 Conversation ChatGPT

## 13.Remerciement

Je voudrais remercier les personnes qui m'ont aidé à réaliser ce travail :

**Mr. Oscar François** pour m'avoir soutenu tout le long du projet.

**Mr. Folly Sodoga** pour m'avoir aidé à résoudre certains problèmes.

**Mr. Sébastien Gaille** pour avoir dessiné les sprites, fond d'écran et pour m'avoir aidé à réaliser la documentation technique. Il m'a aussi encouragé à rester motivé tout au long de mon projet.

## 14. Code source

### Annexe 1

Code source du logiciel

```
1 <?php
2 /**
3  * Description: Page principale
4 */
5 require_once(__DIR__ . '/model/functions/user.php');
6 // Vérifie si l'utilisateur est connecté
7 if (!isset($_SESSION['nickname'])) {
8     header("Location: pages/login.php");
9     die();
10}
11if (isset($_GET["logout"])) {
12    logOut();
13}
14?>
15<!DOCTYPE html>
16<html lang="fr">
17
18<head>
19    <meta charset="UTF-8">
20    <meta http-equiv="X-UA-Compatible" content="IE=edge">
21    <meta name="viewport" content="width=device-width, initial-scale=1.0">
22    <link rel="stylesheet" href="./css/main.css">
23    <title>Home</title>
24</head>
25
26<body class="game-background" onload="GetAllFood()">
27    <section class="container">
28        <p class="title">Pseudo: <?= $_SESSION["nickname"] ?></p>
29        <a href=".//pages/account.php" class="link">
30            <button class="account">Mon compte</button>
31        </a>
32        <a href=".//pages/categories.php" class="link"><button class="btn">Jouer</button>
33        <a href=".//pages/games.php" class="link"><button class="btn">Rejoindre une
partie</button></a>
34        <a href=".//pages/rules.php" class="link"><button class="btn">Règles</button></a>
35        <a href=".//index.php?logout" class="link"><button class="btn">Déconnexion</button>
36    </section>
37    <script src=".//js/getFoods.js"></script>
38</body>
39</html>
```

## api\getFoods.php

```
1 <?php
2 require_once('.../model/functions/food.php');
3
4 switch ($_SERVER['REQUEST_METHOD']) {
5     case 'GET':
6         methodGET();
7         break;
8 }
9 /**
10 * fonction qui récupère tous les aliments et les encode en JSON
11 *
12 * @return json // liste d'aliments en format JSON
13 */
14 function methodGET()
15 {
16     $foods = GetAllFood();
17     echo json_encode([
18         'foods' => $foods,
19     ], JSON_UNESCAPED_UNICODE);
20 }
```

## api\getUsers.php

```
1 <?php
2 require_once('.../model/functions/functions.php');
3
4 switch ($_SERVER['REQUEST_METHOD']) {
5     case 'GET':
6         UpdateUserInformations();
7         break;
8 }
9
10 /**
11 * fonction qui récupère et encode en JSON toutes les données des utilisateurs présent dans
12 * la partie
13 * @return json // les données des utilisateurs présent dans une partie
14 */
15 function UpdateUserInformations(){
16     if($id = filter_input(INPUT_GET, 'parentUserId', FILTER_VALIDATE_INT)){
17         $users = GetAllUserInformations($id);
18         echo json_encode([
19             'users' => $users
20         ],JSON_UNESCAPED_UNICODE);
21     }
22 }
```

## controller\gameController.php

```
1 <?php
2
3 /**
4  * Description : Page qui contient tout le code php de la page game
5 */
6 $date = date('Y-m-d H:i:s');
7
8 // Vérifie si l'utilisateur est connecté
9 if (!isset($_SESSION['nickname'])) {
10     header("Location: ./login.php");
11     die();
12 }
13
14 // On vérifie si la variable de session existe
15 if (!isset($_GET["category"])) {
16     // Rédirection
17     header("Location:./categories.php");
18     die();
19 } else {
20     if (!isset($_GET["date_last_update"]) && !isset($_GET["date_start"]) &&
21     !isset($_GET["idGame"])) {
22         if (isUserInGame($_SESSION["idUtilisateur"]) == 0) {
23             // On vérifie si l'utilisateur arrive dupuis la page categories
24             if (isset($_GET["startGame"])) {
25                 unset($_GET["startGame"]);
26                 // Création d'une nouvelle partie
27                 CreateGame($date, 0, 0, 0, $_SESSION['idUtilisateur'], true,
28 $GET["category"], $_SESSION['idUtilisateur']);
29             }
30         }
31     } else {
32         if (isset($_GET["startGame"])) {
33             unset($_GET["startGame"]);
34             // On vérifie si le joueur arrive de la page games
35             if (JoinGame(
36                 $_GET["date_start"],
37                 $_GET["date_last_update"],
38                 $_GET['score'],
39                 $_GET['slice_count'],
40                 $_GET["duration"],
41                 $_SESSION['idUtilisateur'],
42                 true,
43                 $_GET["category"],
44                 $_GET['parentUserId']
45             ) != null) {
46                 // Si la fonction retourne quelque chose on stocke le message d'erreur dans
47                 // la variable $message
48                 $message = JoinGame(
49                     $_GET["date_start"],
50                     $_GET["date_last_update"],
51                     $_GET['score'],
52                     $_GET['slice_count'],
53                     $_GET["duration"],
54                     $_SESSION['idUtilisateur'],
55                     true,
56                     $_GET["category"],
57                     $_GET['parentUserId']
58             )
59         }
60     }
61 }
```

```
55      );
56      // Redirection vers la page games.php avec le message d'erreur
57      header("Location:./games.php?message=$message");
58      die();
59  }
60 }
61 }
62 }
63 // On vérifie si la requête HTTP a été reçue par la page
64 if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) &&
strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest') {
65
66 if (isset($_POST['data'])) {
67
68     $receivedData = json_decode($_POST['data']);
69     // Si oui on met à jour les données de la partie
70     UpdateGame($receivedData[0], $receivedData[1], $_SESSION['idUtilisateur'],
$receivedData[2]);
71 } else {
72     echo "Aucune donnée reçue.";
73 }
74 exit;
75 }
76 }
```

## controller\inscriptionsController.php

```
1 <?php
2
3 /**
4  * Description : Page qui contient tout le code php de la page inscriptions
5 */
6
7 // On filtre les champs nickname et password
8 $nickname = filter_input(INPUT_POST, "nickname", FILTER_UNSAFE_RAW);
9 $password = filter_input(INPUT_POST, "password", FILTER_UNSAFE_RAW);
10 $btn = filter_input(INPUT_POST, "btn");
11 $message = "";
12 // Vérifie si l'utilisateur est connecté
13 if (isset($_SESSION["nickname"])) {
14     header("Location: ../index.php");
15     die();
16 }
17 //On vérifie si l'utilisateur a appuyer sur le bouton
18 if ($btn != null) {
19     if ($nickname != null && $password != null) {
20         if (addUser($nickname, $password, date('Y-m-d H:i:s')) == null) {
21             // Redirection vers la page login avec le message
22             header("Location: login.php?message=Votre compte vient d'être créé");
23             die();
24         } else {
25             // On stocke le message d'erreur
26             $message = addUser($nickname, $password, date('Y-m-d H:i:s'));
27         }
28     } else {
29         // On stocke le message d'erreur si un ou les deux champs sont vides
30         $message = "Veuillez remplir tous les champs";
31     }
32 }
33 }
```

## controller\loginController.php

```
1 <?php
2
3 /**
4  * Description : Page qui contient tout le code php de la page login
5 */
6
7 // On filtre les champs nickname et password
8 $nickname = filter_input(INPUT_POST, "nickname", FILTER_UNSAFE_RAW);
9 $password = filter_input(INPUT_POST, "password", FILTER_UNSAFE_RAW);
10 $btn = filter_input(INPUT_POST, "btn");
11 $message = "";
12 // Vérifie si l'utilisateur est connecté
13 if (isset($_SESSION["nickname"])) {
14     header("Location: ../index.php");
15     die();
16 }
17 if (isset($_GET["message"])) {
18     $message = $_GET["message"];
19 }
20 if ($btn != null) {
21     if ($nickname != null && $password != null) {
22         if (LoginUser($nickname, $password)) {
23             // Redirection vers la page index.php si la connection a fonctionné
24             header("Location: ../index.php");
25         } else {
26             // On stocke le message d'erreur si un ou des champs sont incorrectes
27             $message = "Un ou des champs sont incorrect";
28         }
29     } else {
30         // On stocke le message d'erreur si un ou les deux champs sont vides
31         $message = "Veuillez remplir tous les champs";
32     }
33 }
34 }
```

## css\main.css

```
1  /*
2  * Description : Page css
3  */
4  *
5  {
6      box-sizing: border-box;
7      user-select: none;
8      -webkit-user-select: none;
9      -khtml-user-select: none;
10     -moz-user-select: none;
11     -o-user-select: none;
12 }
13
14 .game-background{
15     background-image: url("../img/background.png");
16     background-size: cover;
17     margin: auto;
18     width: 95vw;
19     height: 95vh;
20 }
21
22 .container{
23     width:50vw;
24     height:80vh;
25     margin:auto;
26 }
27
28 .btn{
29     width: 50%;
30     height:10%;
31     background-color: #04AA6D;
32     font-family: comic sans ms;
33     color: white
34     cursor: pointer;
35     display: block;
36     margin-top: 5%;
37     margin-left: 20%;
38     border-style: none;
39     font-size: 1.5vw;
40     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color 0.5s, font-size 0.5s;
41 }
42
43 .btn:hover{
44     width: 35%;
45     height:10%;
46     background-color: #04AA6D;
47     font-family: comic sans ms;
48     color: white
49     cursor: pointer;
50     display: block;
51     margin-top: 5%;
52     margin-left: 20%;
53     border-style: none;
54     font-weight: bold;
55     font-size: large;
```

```
56     font-size: 1.5vw;
57     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color
58     0.5s, font-size 0.5s;
59 }
59
60 .btn-game-over{
61     width: 50%;
62     height:10%;
63     background-color: #04AA6D;
64     font-family: comic sans ms;
65     color: white
66     cursor: pointer;
67     border-style: none;
68     font-size: 1.5vw;
69     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color
70     0.5s, font-size 0.5s;
71 }
71
72 .btn-game-over:hover{
73     width: 35%;
74     height:10%;
75     background-color: #04AA6D;
76     font-family: comic sans ms;
77     color: white
78     cursor: pointer;
79     border-style: none;
80     font-weight: bold;
81     font-size: large;
82     font-size: 1.5vw;
83     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color
84     0.5s, font-size 0.5s;
85 }
85
86 .pseudo{
87     font-size: 2vw;
88     color:white
89     margin-left: 20%;
90     font-family: comic sans ms;
91 }
92
93 .title{
94     font-size: 1.5vw;
95     color:white
96     margin-left: 20%;
97     width:50%;
98     font-family: comic sans ms;
99     text-align: center;
100    margin-left:23%;
101 }
102
103 .account{
104     font-size: 1.5vw;
105     color:white
106     width:50%;
107     height:10%;
108     font-family: comic sans ms;
109     text-align: center;
110     font-weight: bold;
111     margin-left:20%;
112     margin-top:1%;
```

```
113     text-decoration: none;
114     background-color: #07b1a3;
115     border:none;
116     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color 0.5s, font-size 0.5s;
117 }
118
119 .account:hover{
120     font-size: 1.5vw;
121     color:white
122     width:40%;
123     height:10%;
124     font-family: comic sans ms;
125     text-align: center;
126     font-weight: bold;
127     margin-left:20%;
128     margin-top:1%;
129     text-decoration: none;
130     background-color: #07b1a3;
131     border:none;
132     font-weight: bold;
133     transition: width 0.5s, height 0.5s, background-color 0.5s, transform 0.5s, color 0.5s, font-size 0.5s;
134 }
135
136
137 .description{
138     font-size: 0.9vw;
139     color:white
140     font-family: comic sans ms;
141     text-align: center;
142 }
143
144
145 .description-h2{
146     color:white
147     font-family: comic sans ms;
148     text-align: center;
149 }
150
151 .games-text{
152     font-size: 1vw;
153     color:white
154     margin-left: 20%;
155     width:50%;
156     font-family: comic sans ms;
157     text-align: center;
158     margin-left:20%;
159 }
160
161 .error-text{
162     font-size: 1vw;
163     color:rgb(86, 1, 1);
164     margin-left: 20%;
165     width:50%;
166     font-family: comic sans ms;
167     text-align: center;
168     margin-left:20%;
169 }
170
```

```
171 .link{
172     text-decoration: none;
173 }
174
175
176 .text{
177     font-family: comic sans ms;
178     color: white
179 }
180
181 .message{
182     color: red
183 }
184
185 .img-rules{
186     margin-top: 2%;
187     margin-left: 25%;
188 }
189 #game-over-menu{
190     visibility: hidden;
191     border: 5px solid black;
192     width: 40vw;
193     height: 40vh;
194     margin-left: 30%;
195     margin-top: 7%;
196     text-align: center;
197 }
198
199 #canvas{
200     margin: auto;
201     position: absolute;
202     border: 5px solid black;
203 }
204
205 #game-over-menu h1{
206     margin-top: 15%;
207     color: white
208 }
209
210 #game-over-menu p{
211     color: white
212 }
213
214 #result{
215     visibility: visible;
216 }
```

## js\addBadFood.js

```
1 /**
2  * fonction qui permet d'ajouter deux aliments malsains dans la liste
3  * @param {*} list // (liste d'aliments)
4  */
5 export function addBadFood(list) {
6     list.push({ category: "malbouffe", name: "Fausse-tomate", picture_path: "Fausse-
7         tomate.png" });
8     list.push({ category: "malbouffe", name: "Fausse-aubergine", picture_path: "Fausse-
9         aubergine.png" });
}
```

**js\game.js**

```

1 /**
2  * Description : Script principale du jeu
3 */
4 import { sendDataToPhp } from "./sendDataToPhp.js";
5 import { addBadFood } from "./addBadFood.js";
6 import { updateUsers } from "./updateUsers.js";
7
8 // Déclarations des variables
9 let nbFood = 1;
10 let speed = 1;
11 let score = 0;
12 let foods = [];
13 let slice_count = 0;
14 let isMultiplayer = false;
15 // Récupération d'aliments
16 let foodList = JSON.parse(localStorage.getItem("Foods"));
17 // Ajout des aliments malsains dans la liste
18 addBadFood(foodList);
19 // Récupération des éléments HTML
20 let speedElement = document.getElementById("speed");
21 let gameOverMenu = document.getElementById("game-over-menu");
22 let gameOverMenuButton = document.getElementById("game-over-menu-button");
23 let informations = document.getElementById("informations");
24 let canvas = document.getElementById('canvas');
25 let ctx = canvas.getContext('2d');
26 // Définition de la taille du canvas
27 canvas.width = window.innerWidth * 0.95;
28 canvas.height = window.innerHeight * 0.75;
29 let FoodWidth = canvas.width / 25;
30 let FoodHeight = canvas.width / 20;
31 let url = new URL(window.location.href);
32 // Récupérations des variables de l'URL
33 let URLcategory = url.searchParams.get("category");
34 let idUser = url.searchParams.get("idUser");
35 speedElement.innerText = "Vitesse: *" + Math.round(speed * 10) / 10;
36
37 canvas.addEventListener('click', handleClickEvent, false);
38 window.addEventListener('resize', resizeCanvas, false);
39
40 let drawInterval = setInterval(draw, 10); // Boucle principale
41 let UpdateInformaions = setInterval(updateUsers, 100);
42 updateUsers();
43
44 /**
45  * fonction qui permet de recuperer un aliment aléatoire de la liste
46  * @param {Array} list la liste des aliments
47  * @returns
48 */
49 function randomFoodFromList(list) {
50 // On filtre les aliments de la catégorie
51 let foodCategorieList = list.filter(item => item.category === URLcategory);
52 if (foodCategorieList.length > 0 && Math.random() < 0.75) {
53 // 75% de chance de tomber sur les aliments de la catégorie choisie
54 return foodCategorieList[Math.floor(Math.random() * foodCategorieList.length)];
55 } else {
56 return list[Math.floor(Math.random() * list.length)];
}

```

```
57  }
58 }
59 /**
60 * fonction qui permet de crée un nouveau aliment
61 * @returns food
62 */
63 function createFood() {
64     let randomFood = randomFoodFromList(foodList);
65     let food = {
66         x: Math.random() * Math.floor(Math.random() * (canvas.width - (FoodHeight) -
FoodWidth)) + FoodWidth, // position x aléatoire
67         y: -FoodHeight,
68         width: FoodWidth,
69         height: FoodHeight,
70         src: "../sprites/" + randomFood.picture_path, // chemin de l'image recuperer de la
fonction randomFood()
71         speed: speed,
72         name: randomFood.name, // le nom de l'aliment recuperer de random de la fonction
randomFood()
73         category: randomFood.category, // la catégorie recuperer de randomFood de la fonction
randomFood()
74     };
75     return food;
76 }
77
78
79 /**
80 * fonction qui permet de dessiner les aliments sur le canvas
81 */
82 function draw() {
83     // Effacer le canvas
84     ctx.clearRect(0, 0, canvas.width, canvas.height);
85
86     // Dessine chaque aliment
87     for (let i = 0; i < foods.length; i++) {
88         let food = foods[i];
89         let img = new Image();
90         img.src = food.src;
91         // Ajout de l'aliment sur le canvas
92         ctx.drawImage(img, food.x, food.y, food.width, food.height);
93
94         // Dessine le nom en dessus de l'aliment
95         ctx.font = "bold 13px Comic Sans MS";
96         ctx.fillText(food.name, food.x + food.width / 3, food.y + food.height * 1.2);
97
98         // Fait tomber l'aliment avec la vitesse
99         food.y += food.speed;
100
101        // Si l'aliment atteint le bas de l'écran
102        if (food.y > canvas.height - food.height) {
103            // On vérifie si l'aliment fais partie de la catégorie
104            if (food.category == URLcategory) {
105                // Si oui on lui enlève les points
106                score -= 2;
107                // Si le score est négatif on le met à zéro
108                if (score < 0) {
109                    score = 0;
110                    gameOver();
111                }
112            }
113        // On envoie les donnés au php
```

```

114     sendDataToPhp(URLcategory, score, slice_count, 1);
115     // On supprime l'aliment
116     foods.splice(i, 1);
117     break;
118 }
119 }
120 // On vérifie si le nombre d'aliments est plus petit que le nombre max des aliments
121 if (foods.length < nbFood) {
122     foods.push(createFood());
123 }
124 }
125 /**
126 * fonction qui permet d'augmenter la vitesse du défilement du joueur qui a le score le
127 plus petit de la partie
128 */
129 function addAndShowUsersSpeed() {
130     updateUsers();
131     let userScore = JSON.parse(sessionStorage.getItem("ListUsers"));
132     if (userScore.length > 1) {
133         isMultiplayer = true;
134         // Si le joueur a le score le plus petit alors la vitesse augmente plus vite
135         if (userScore[0]["idUser"] == idUser) {
136             speed += 0.1;
137             speedElement.innerText = "Vitesse: *" + Math.round(speed * 10) / 10;
138         }
139     }
140 }
141 /**
142 * fonction qui permet de vérifier si le cursor est à l'intérieur de l'image
143 * @param {int} x position x du cursor
144 * @param {int} y position y du cursor
145 * @param {array} rect l'aliment
146 * @returns
147 */
148 function isPointerInsideFood(x, y, rect) {
149     return x >= rect.x && x <= rect.x + rect.width &&
150         y >= rect.y && y <= rect.y + rect.height;
151 }
152
153
154 /**
155 * fonction qui permet de gérer de vérifier si un aliment a été clicker
156 * @param {} event
157 */
158 function handleClickEvent(event) {
159
160     // Les coordonnées exacts du curseur
161     let clicX = event.offsetX;
162     let clicY = event.offsetY;
163
164     // Vérifier si le cursor est à l'intérieur d'un aliment
165     for (let i = 0; i < foods.length; i++) {
166         let food = foods[i];
167         if (isPointerInsideFood(clicX, clicY, food)) {
168             // Si l'aliment fait partie de la catégorie on augmente la vitesse, le score du
joueur
169             if (food.category == URLcategory) {
170                 speed += 0.1;
171                 score++;

```

```

172     nbFood = Math.floor(1 + (0.2 * (speed - 1) / 0.1));
173     speedElement.innerText = "Vitesse: *" + Math.round(speed * 10) / 10;
174     slice_count++;
175     addAndShowUsersSpeed();
176     sendDataToPhp(URLcategory, score, slice_count, 1);
177 } else {
178     // Si l'aliment fait partie de la catégorie malbouffe on arrête le jeu
179     if (food.category == "malbouffe") {
180         score = 0;
181         gameOver();
182     }
183     score -= 2;
184     slice_count++;
185     addAndShowUsersSpeed();
186     if (score < 0) {
187         score = 0;
188         gameOver();
189     }
190 }
191 sendDataToPhp(URLcategory, score, slice_count, 1);
192 // Supprimer l'aliment du tableau
193 foods.splice(i, 1);
194 break;
195 }
196 }
197 }
198
199 /**
200 * fonction qui permet de rezise le canvas en fonction de la taille de la fénetre
201 */
202 function resizeCanvas() {
203     canvas.width = window.innerWidth * 0.9;
204     canvas.height = window.innerHeight * 0.9;
205 }
206
207 /**
208 * événement qui permet de vérifier si le joueur a quitté ou rechargé la page
209 */
210 window.addEventListener('beforeunload', function (event) {
211     sendDataToPhp(URLcategory, score, slice_count, 0);
212 });
213
214 /**
215 * fonction qui permet d'afficher le menu de la fin du jeu
216 */
217
218 function gameOver() {
219     let userScore = JSON.parse(sessionStorage.getItem("ListUsers"));
220     // Supprime la variable de l'url
221     url.searchParams.delete("category");
222     history.replaceState(null, "", url);
223     sendDataToPhp(URLcategory, score, slice_count, 0);
224     clearInterval(UpdateInformaions);
225     clearInterval(drawInterval);
226     canvas.remove();
227     // On vérifie si le nombres de joueurs de la partie
228     if (userScore.length < 2) {
229         // On vérifie si le jeu est en multijoueur
230         if (isMultiplayer) {
231             // Affichage du gagnant de la partie

```

```
232     document.getElementById("winner").innerText = "Vous avez gagné la partie!";
233 }
234 }
235 // Affichage de la durée de la partie
236 document.getElementById("duration").innerText =
237 document.getElementById("time").textContent;
238 gameOverMenu.style.visibility = "visible";
239 informations.style.visibility = "hidden";
240 gameOverMenuButton.addEventListener('click', function (event) {
241     location.href = "./categories.php";
242 });
243
244
245
246
247
248 }
```

## js\getFoods.js

```
1  /**
2   * fonction qui permet de stocker tous les aliments dans le localStorage
3   */
4  function GetAllFood() {
5      const config = {
6          method: "get",
7      };
8
9      fetch("./api/getFoods.php", config)
10         .then(response => { return response.json() })
11         .then(json => {
12             // On stocke la liste des aliments dans une variable localstorage
13             localStorage.setItem("Foods", JSON.stringify(json.foods));
14         });
15 }
```

## js\sendDataToPhp.js

```
1 /**
2  * fonction qui permet d'envoyer les données javascript en php via une requête HTTP
3  * @param {string} categorie // (catégorie d'aliments que le joueur a choisi)
4  * @param {int} score // (le score du joueur)
5  * @param {int} slice_count // (nombres d'aliments que le joueur a découpé)
6  * @param {bool} status // (partie terminé ou pas)
7  */
8
9 export function sendDataToPhp(categorie, score, slice_count, status) {
10     // Les données à envoyé
11     let dataToSend = JSON.stringify([score, slice_count, status]);
12     fetch('../pages/game.php?category=' + categorie, {
13         method: 'POST',
14         headers: {
15             // On indique les headers
16             'Content-Type': 'application/x-www-form-urlencoded',
17             'X-Requested-With': 'XMLHttpRequest'
18         },
19         // On envoie les données
20         body: 'data=' + encodeURIComponent(dataToSend)
21     })
22     .then(response => response.text())
23     .then(result => {
24         document.getElementById("result").innerHTML = result;
25     })
26     // S'il y a une erreur on l'affiche dans la console
27     .catch(error => {
28         console.error('Erreur:', error);
29     });
30 }
```

## js\updateUsers.js

```

1  /**
2   * fonction qui permet de mettre à jour les informations de tous les joueurs présent dans
3   * une partie
4  */
5  export function updateUsers() {
6      const config = {
7          method: "get",
8      };
9      var url = new URL(document.location.href);
10     var search_params = new URLSearchParams(url.search);
11     // On vérifie si le paramètre id existe dans l'url
12     if (search_params.has('parentUserId')) {
13         var id = search_params.get('parentUserId');
14         // On récupere le résultat que retourne l'api getUsers
15         fetch('../api/getUsers.php?parentUserId=' + id, config)
16             .then(response => { return response.json() })
17             .then(json => {
18                 let nicknames = "";
19                 let time = ""
20                 json.users.forEach(element => {
21                     nicknames += "Joueur: " + element["nickname"] + " Score: " +
element["score"] + "\n";
22                     time = element["date_start"];
23                 });
24                 // On affiche les données des utilisateurs
25                 document.getElementById("nickaname").innerText = nicknames;
26                 let now = new Date();
27                 let date_start = new Date(time);
28
29                 // Calculer la différence en millisecondes entre la date de début de la
partie et la date actuelle
30                 let difference = now.getTime() - date_start.getTime();
31
32                 // Convertir la différence en millisecondes en secondes, minutes, heures
33                 let seconds = Math.floor(difference / 1000);
34                 let minutes = Math.floor(seconds / 60);
35                 let hours = Math.floor(minutes / 60);
36
37                 // Calculer le temps restant dans chaque unité
38                 let GameSeconds = seconds % 60;
39                 let GameMinutes = minutes % 60;
40                 let GameHours = hours % 24;
41
42                 // On affiche le temps écoulé depuis le début de la partie
43                 document.getElementById("time").innerText = "Temps : " + GameHours + ":" +
GameMinutes + ":" + GameSeconds;
44                 sessionStorage.setItem('ListUsers', JSON.stringify(json.users));
45             });
46     }
47 }
```

## js\user.js

```
1  /**
2   * fonction qui permet d'afficher une popup qui demande à l'utilisateur s'il veut vraiment
3   * supprimer son compte
4   */
5  function DeleteUserYesOrNo(){
6      const response = confirm("Voulez-vous vraiment supprimer ce compte?");
7      if (response) {
8          location.href = "../pages/account.php?delete";
9      }
10
11
12
13 }
```

## model\conf.php

```
1 <?php
2 /**
3  * Description: Page qui contient les données pour se connecter à la base de donné
4 */
5 define( "DB_HOST",      "localhost");
6 define( "DB_NAME",      "db_ninja");
7 define( "DB_USER",      "slim_ninja_user");
8 define( "DB_PASSWORD",  "TPI2023");
```

model\myDB.php

```
1 <?php
2
3 /**
4  * Description : Page de connection à la base de donnée
5 */
6
7 require_once("conf.php");
8 /**
9  * fonction qui se connecte à la base de données
10 *
11 * @param string sql
12 * @param string param
13 * @return object
14 */
15 function db()
16 {
17     try {
18         static $myDb = null;
19         if ($myDb == null) {
20             $myDb = new PDO(
21                 "mysql:host=" . DB_HOST . ";dbname=" . DB_NAME . ";charset=utf8",
22                 DB_USER,
23                 DB_PASSWORD
24             );
25             $myDb->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
26             $myDb->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
27         }
28         return $myDb;
29     } catch (Exception $e) {
30         die("services momentanément indisponible. Merci de réessayer plus tard");
31     }
32 }
33
34 /**
35  * fonction qui exécute une requête SQL
36 *
37 * @param string sql
38 * @param string param
39 * @return object
40 */
41 function dbRun($sql, $param = null)
42 {
43     $statement = db()->prepare($sql);
44
45     $statement->execute($param);
46
47     return $statement;
48 }
49
50
51
52
53 }
```

## model\functions\food.php

```

1 <?php
2 require_once __DIR__ . '/../myDB.php';
3 /**
4  * fonction qui permet de récupérer la liste des aliments
5  *
6  * @return array
7  */
8 function GetAllFood()
9 {
10    $sql = "SELECT category,name,picture_path from item";
11    $param = [];
12    $statement = dbRun($sql, $param);
13    return $statement->fetchAll(PDO::FETCH_ASSOC);
14 }
15 /**
16  * fonction qui permet de créer une nouvelle partie
17  *
18  * @param DateTime $date_start (date de début de la partie)
19  * @param int $score
20  * @param int $slice_count (nombres d'aliments découpé)
21  * @param int $duration (la durée de la partie)
22  * @param int $idUser (l'id de l'utilisateur)
23  * @param bool $status (partie terminé ou pas)
24  * @param string $category (la catégorie d'aliments que le joueur a choisi)
25  * @param int $parentIdUser (l'id du joueur qui a créé la partie)
26  * @return void
27  */
28 function CreateGame(
29    $date_start,
30    $score,
31    $slice_count,
32    $duration,
33    $idUser,
34    $status,
35    $category,
36    $parentIdUser
37 ) {
38    $sql = "INSERT INTO `game`(
39        `date_start`, `date_last_update`, `score`, `slice_count`, `duration`, `idUser`,
40        `status`, `category`, `parentIdUser`) value
41        (:date_start,:date_last_update,:score,:slice_count,:duration,:idUser,
42        :status,:category,:parentIdUser)";
43    $param = [
44        'date_start' => $date_start,
45        'date_last_update' => $date_start,
46        'score' => $score,
47        'slice_count' => $slice_count,
48        'duration' => $duration,
49        'idUser' => $idUser,
50        'status' => $status,
51        'category' => $category,
52        'parentIdUser' => $parentIdUser,
53    ];
54    dbRun($sql, $param);
55 }
56

```

```

57 /**
58 * fonction qui permet de mettre à jour les informations d'une partie déjà en cours
59 *
60 * @param int $score
61 * @param int $slice_count (nombres d'aliments découpé)
62 * @param int $idUser (l'id de l'utilisateur)
63 * @param bool $status (partie terminé ou pas)
64 * @return void
65 */
66 function UpdateGame($score, $slice_count, $idUser, $status)
67 {
68     $sql = "UPDATE `game` SET `date_last_update` = NOW(),
69         `score`=:score, `slice_count` = :slice_count,
70         `duration` = TIMEDIFF(date_last_update, date_start), `status` = :status
71     WHERE idUser = :idUser AND status = true";
72     $param = [
73         'score' => $score,
74         'slice_count' => $slice_count,
75         'status' => $status,
76         'idUser' => $idUser,
77     ];
78     dbRun($sql, $param);
79 }
80
81 /**
82 * fonction qui permet de récupérer toutes les parties en cours
83 *
84 * @param $idUser (l'id de l'utilisateur)
85 * @return array
86 */
87 function GetOnGoingGames($idUser)
88 {
89
90     $sql = "SELECT game.idGame, game.category, game.date_start, game.date_last_update,
91             game.score, game.slice_count, game.duration, game.parentIdUser,
92             user.nickname FROM game
93             INNER JOIN user ON game.idUser = user.idUser
94             where game.status = true and game.parentIdUser != :parentIdUser and
95             game.idUser != :idUser";
96     $param = [
97         'parentIdUser' => $idUser,
98         'idUser' => $idUser,
99     ];
100    $statement = dbRun($sql, $param);
101    return $statement->fetchAll(PDO::FETCH_ASSOC);
102 }
103 /**
104 * fonction qui permet de vérifier si un joueur n'est pas déjà dans une partie en cours
105 *
106 * @param int $idUser (l'id de l'utilisateur)
107 * @return boolean
108 */
109 function isUserInGame($idUser)
110 {
111     $sql = "SELECT COUNT(idUser) from game where idUser = :idUser and status = true
112             and parentIdUser=:parentIdUser";
113     $param = [
114         'idUser' => $idUser,
115         'parentIdUser' => $idUser,
116     ];

```

```

117     $statement = dbRun($sql, $param)->fetch(PDO::FETCH_ASSOC);
118     return $statement["COUNT(idUser)"];
119 }
120
121 /**
122 * fonction qui récupère le nombre de joueurs présents dans une partie
123 *
124 * @param $parentIdUser (l'id du joueur qui a créé la partie)
125 * @param $date_start (date de début de la partie)
126 * @return int
127 */
128 function nbPlayersInGame($parentIdUser, $date_start)
129 {
130     $sql = "SELECT COUNT(parentIdUser) from game where parentIdUser = :parentIdUser and
131     status = true and
132     date_start = :date_start";
133     $param = [
134         'parentIdUser' => $parentIdUser,
135         'date_start' => $date_start,
136     ];
137     $statement = dbRun($sql, $param)->fetch(PDO::FETCH_ASSOC);
138     return $statement["COUNT(parentIdUser)"];
139 }
140
141 /**
142 * fonction qui permet de créer une nouvelle partie pour un joueur qui veut rejoindre une
143 * partie déjà en cours
144 *
145 * @param DateTime $date_start (date de début de la partie)
146 * @param DateTime $date_last_update (date de la dernière mise à jour de la partie)
147 * @param int $score
148 * @param int $slice_count (nombres d'aliments découpé)
149 * @param int $duration (la durée de la partie)
150 * @param int $idUser (l'id de l'utilisateur)
151 * @param string $status (partie terminée ou pas)
152 * @param string $category (la catégorie d'aliments que le joueur a choisi)
153 * @return void || string
154 */
155 function JoinGame(
156     $date_start,
157     $date_last_update,
158     $score,
159     $slice_count,
160     $duration,
161     $idUser,
162     $status,
163     $category,
164     $parentIdUser
165 ) {
166     if (nbPlayersInGame($parentIdUser, $date_start) != 2) {
167         if (isUserInGame($idUser) < 2) {
168             $sql = "INSERT INTO `game`"
169             (`date_start`, `date_last_update`, `score`, `slice_count`, `duration`,
170             `idUser`, `status`, `category`, `parentIdUser`)
171             value
172             (:date_start,:date_last_update,:score,:slice_count,:duration,
173             :idUser,:status,:category,:parentIdUser)";
174             $param = [
175                 'date_start' => $date_start,
176                 'date_last_update' => $date_last_update,

```

```
175         'score' => $score,
176         'slice_count' => $slice_count,
177         'duration' => $duration,
178         'idUser' => $idUser,
179         'status' => $status,
180         'category' => $category,
181         'parentIdUser' => $parentIdUser,
182     ];
183     dbRun($sql, $param);
184 } else {
185     return "Vous avez déjà rejoint cette partie!";
186 }
187 } else {
188     return "La partie est pleine!";
189 }
190 }
191 /**
192 * fonction qui permet de récupérer toutes les informations des joueurs présents dans une
193 partie et trient les joueurs selon leurs scores
194 *
195 * @param $parentIdUser (l'id du joueur qui a crée la partie)
196 * @return array
197 */
198 function GetAllUserInformations($parentIdUser)
199 {
200     $sql = "SELECT
201 game.score,game.idUser,game.slice_count,game.date_start,game.duration,user.nickname
202     from game INNER JOIN user ON game.idUser = user.idUser
203     where game.parentIdUser = :parentIdUser and game.status = true order by game.score
204 asc";
205     $param = [
206         'parentIdUser' => $parentIdUser,
207     ];
208     $statement = dbRun($sql, $param)->fetchAll(PDO::FETCH_ASSOC);
209     return $statement;
}
```

## model\functions\user.php

```

1 <?php
2 session_start();
3
4 require_once __DIR__ . '/..../myDB.php';
5 // La date est mise à l'heure de Zurich
6 date_default_timezone_set('Europe/Zurich');
7
8
9 /**
10 * fonction qui permet la connexion d'un utilisateur
11 *
12 * @param string $nickname (pseudo du joueur)
13 * @param string $pwd (mot de passe du joueur)
14 * @return bool
15 */
16 function LoginUser($nickname, $pwd)
17 {
18     $conn = db();
19     // Vérification de l'email
20     $query = $conn->prepare("SELECT * FROM user WHERE `nickname` = :nickname;");
21     $query->bindParam(":nickname", $nickname, PDO::PARAM_STR);
22     $query->execute();
23     $result = $query->fetch(PDO::FETCH_ASSOC);
24     if ($result !== null && $result != "" && $result !== false) {
25         // Vérification du mot de passe
26         if (password_verify($pwd, $result["password"])) {
27             // création des variables de session une fois que l'utilisateur est connecté
28             $_SESSION['idUtilisateur'] = $result["idUser"];
29             $_SESSION['nickname'] = $result["nickname"];
30             return true;
31         }
32     }
33 }
34
35 /**
36 * fonction qui permet de récupérer un utilisateur selon son nom
37 *
38 * @param string $nickname (pseudo du joueur)
39 * @return array
40 */
41 function selectUserbyNickname($nickname)
42 {
43     $conn = db();
44     // Vérification de l'email
45     $query = $conn->prepare("SELECT * FROM user WHERE `nickname` = :nickname;");
46     $query->bindParam(":nickname", $nickname, PDO::PARAM_STR);
47     $query->execute();
48     $result = $query->fetch(PDO::FETCH_ASSOC);
49     return $result;
50 }
51
52 /**
53 * fonction qui permet d'inscrire un utilisateur
54 *
55 * @param string $nickname (pseudo du joueur)
56 * @param string $hash_pass (mot de passe haché du joueur)

```

```
57 * @param DateTime $registration_date (date d'inscription du joueur)
58 * @return void || string
59 */
60 function addUser($nickname, $hash_pass, $registration_date)
61 {
62     $pdo = db();
63     if (selectUserbyNickname($nickname) == null) {
64         $query = $pdo->prepare("INSERT INTO `user`"
65             (`nickname`, `password`, `registration_date`)VALUES(?, ?, ?)");
66         $query->execute([$nickname, password_hash($hash_pass, PASSWORD_DEFAULT),
$registration_date]);
67     } else {
68         return "Pseudo déjà utilisé";
69     }
70 }
71 /**
72 * fonction qui permet la déconnexion d'un utilisateur
73 *
74 * @return void
75 */
76 function LogOut()
77 {
78     // Supprime toutes les variables de session
79     $_SESSION = [];
80     // Supprime la session
81     session_destroy();
82     // Redirige l'utilisateur vers la page home
83     header("Location: ./index.php");
84     die();
85 }
86
87 /**
88 * fonction qui permet de supprimer un compte utilisateur
89 *
90 * @param int $idUser (l'id du joueur)
91 * @return void
92 */
93 function DeleteUser($idUser)
94 {
95     $sql = "DELETE from user where idUser = :idUser";
96     $param = [
97         'idUser' => $idUser,
98     ];
99     dbRun($sql, $param);
100    $_SESSION = [];
101    // Supprime la session
102    session_destroy();
103 }
```

pages\account.php

```
1 <?php
2 /**
3  * Description: Page de l'utilisateur
4 */
5 require_once('../model/functions/user.php');
6 // Vérifie si l'utilisateur est connecté
7 if (!isset($_SESSION['nickname'])) {
8     header("Location: ./login.php");
9 }
10 if (isset($_GET["delete"])) {
11     DeleteUser($_SESSION["idUtilisateur"]);
12     header("Location: ./login.php");
13     die();
14 }
15 ?>
16 <!DOCTYPE html>
17 <html lang="fr">
18 <head>
19     <meta charset="UTF-8">
20     <meta http-equiv="X-UA-Compatible" content="IE=edge">
21     <meta name="viewport" content="width=device-width, initial-scale=1.0">
22     <link rel="stylesheet" href="../css/main.css">
23     <title>Account</title>
24 </head>
25
26 <body class="game-background">
27     <section class="container">
28         <p class="title">Pseudo: <?= $_SESSION["nickname"] ?></p>
29         <a href="../index.php" class="link"><button class="account">Retour <<< </button>
30             <button class="btn" onclick="DeleteUserYesOrNo()">Supprimer mon compte</button>
31         </a>
32         </section>
33         <script src="../js/user.js"></script>
34     </body>
35 </html>
```

## pages\categories.php

```
1 <?php
2 /**
3  * Description: Page des catégories
4 */
5 require_once('../model/functions/user.php');
6 require_once('../model/functions/food.php');
7 // Vérifie si l'utilisateur est connecté
8 if(!isset($_SESSION['nickname'])){
9     header("Location: ./login.php");
10 }
11 ?>
12 <!DOCTYPE html>
13 <html lang="fr">
14 <head>
15     <meta charset="UTF-8">
16     <meta http-equiv="X-UA-Compatible" content="IE=edge">
17     <meta name="viewport" content="width=device-width, initial-scale=1.0">
18     <link rel="stylesheet" href="../css/main.css">
19     <title>Categories</title>
20 </head>
21 <body class="game-background">
22     <section class="container">
23         <a href="../pages/account.php" class="link"><p class="title">Pseudo: <?=
24             $_SESSION["nickname"]?></p></a>
25         <a href="../index.php" class="link"><button class="account">Retour <<</button></a>
26         <a href="game.php?category=fruit&startGame=true&parentUserId=<?=
27             $_SESSION["idUtilisateur"]?>&idUser=<?=$_SESSION["idUtilisateur"]?>" class="link"><button
28             class="btn">Fruit</button></a>
29         <a href="game.php?category=légume&startGame=true&parentUserId=<?=
30             $_SESSION["idUtilisateur"]?>&idUser=<?=$_SESSION["idUtilisateur"]?>" class="link"><button
31             class="btn">Légume</button></a>
32         <a href="game.php?category=légumineuse&startGame=true&parentUserId=<?=
33             $_SESSION["idUtilisateur"]?>&idUser=<?=$_SESSION["idUtilisateur"]?>" class="link"><button
34             class="btn">Légumineuse</button></a>
35         <a href="game.php?category=céréale&startGame=true&parentUserId=<?=
36             $_SESSION["idUtilisateur"]?>&idUser=<?=$_SESSION["idUtilisateur"]?>" class="link"><button
37             class="btn">Céréale</button></a>
38         <a href="game.php?category=féculent&startGame=true&parentUserId=<?=
39             $_SESSION["idUtilisateur"]?>&idUser=<?=$_SESSION["idUtilisateur"]?>" class="link"><button
40             class="btn">Féculent</button></a>
41     </section>
42
43 </body>
44 </html>
```

## pages\game.php

```
1 <?php
2 /**
3  * Description: Page du jeu
4 */
5 require_once('../model/functions/user.php');
6 require_once('../model/functions/food.php');
7 require_once('../controler/gameControler.php');
8 ?>
9 <!DOCTYPE html>
10 <html lang="fr">
11 <head>
12     <meta charset="UTF-8">
13     <meta http-equiv="X-UA-Compatible" content="IE=edge">
14     <meta name="viewport" content="width=device-width, initial-scale=1.0">
15     <link rel="stylesheet" href="../css/main.css">
16     <title>Game</title>
17 </head>
18 <body class="game-background">
19     <div id="informations">
20         <p id="nickaname" class="text">Joueurs : <?= $_SESSION['nickname']; ?></p>
21         <p class="text" id="speed">Vitesse :</p>
22         <p class="text" id="time">Temps :</p>
23     </div>
24     <canvas id="canvas" ></canvas>
25     <div id="result"></div>
26     <div id="game-over-menu">
27         <h1>Vous avez perdu!</h1>
28         <p id="duration"></p>
29         <p id="winner"></p>
30         <button id="game-over-menu-button" class="btn-game-over">Retour</button>
31     </div>
32     <script src="../js/game.js" type="module"></script>
33     <script src="../js/sendDataToPhp.js" type="module"></script>
34     <script src="../js/addBadFood.js" type="module" ></script>
35     <script src="../js/updateUsers.js" type="module"></script>
36 </body>
37 </html>
```

## pages\games.php

```

1 <?php
2 /**
3  * Description: Page des parties en cours
4 */
5 require_once('../model/functions/user.php');
6 require_once('../model/functions/food.php');
7 // Vérifie si l'utilisateur est connecté
8 if (!isset($_SESSION['nickname'])) {
9     header("Location: ./login.php");
10}
11//Récupération des parties
12$games = GetOnGoingGames($_SESSION["idUtilisateur"]);
13?>
14<!DOCTYPE html>
15<html lang="fr">
16
17<head>
18    <meta charset="UTF-8">
19    <meta http-equiv="X-UA-Compatible" content="IE=edge">
20    <meta name="viewport" content="width=device-width, initial-scale=1.0">
21    <link rel="stylesheet" href="../css/main.css">
22    <title>Games</title>
23</head>
24<body class="game-background">
25    <section class="container">
26        <a href="../pages/account.php" class="link">
27            <p class="title">Pseudo: <?= $_SESSION["nickname"] ?></p>
28        </a>
29        <a href="../index.php" class="link"><button class="account">Retour <<< </button>
30        <!--Affichage des parties-->
31        <?php foreach ($games as $game) {
32        ?>
33            <p class="games-text">Hôte: <?php echo $game["nickname"]; ?></p>
34            <p class="games-text">Categorie: <?php echo $game["category"]; ?></p>
35            <a href="game.php?startGame=true&category=<?= $game["category"] ?>&date_start=<?= $game["date_start"] ?>
36                &date_last_update=<?= $game["date_last_update"] ?>&idGame=<?= $game["idGame"] ?>&score=<?= $game["score"] ?>
37                &slice_count=<?= $game["slice_count"] ?>&duration=<?= $game["duration"] ?>
38                &parentUserId=<?= $game["parentIdUser"] ?>
39                &idUser=<?= $_SESSION["idUtilisateur"] ?>" class="link"><button
40                class="btn">Rejoindre</button></a>
41        <?php }
42        if (isset($_GET['message'])) {
43            echo "<p class='error-text'>" . $_GET['message'] . "</p>";
44        }
45    ?>
46    </section>
47</body>
48
49</html>

```

## pages\inscriptions.php

```
1 <?php
2 /**
3  * Description: Page d'inscription
4 */
5 require_once('../model/functions/user.php');
6 require_once('../controler/inscriptionsControler.php');
7 ?>
8 <!DOCTYPE html>
9 <html lang="fr">
10
11 <head>
12     <meta charset="UTF-8">
13     <meta http-equiv="X-UA-Compatible" content="IE=edge">
14     <meta name="viewport" content="width=device-width, initial-scale=1.0">
15     <link rel="stylesheet" href="../css/main.css">
16     <title>Inscription</title>
17 </head>
18
19 <body>
20     <h1>Inscription:</h1>
21     <form method="post" action="">
22         <p>Pseudo:</p>
23         <input type="text" minlength="3" maxlength="15" name="nickname" placeholder="(15
24 caractères max)" required>
25         <p>Mot de passe:</p>
26         <input type="password" minlength="3" name="password" placeholder="mot de passe"
27 required>
28         <input type="submit" value="valider" name="btn">
29         <a href=".//login.php"></p>J'ai déjà un compte</p></a>
30         <p class="message"><?= $message; ?></p>
31     </form>
32 </body>
33
34 </html>
```

## pages\login.php

```
1 <?php
2 /**
3  * Description: Page de la connexion
4 */
5 require_once('../model/functions/user.php');
6 require_once('../controler/loginControler.php');
7 ?>
8 <!DOCTYPE html>
9 <html lang="fr">
10
11 <head>
12     <meta charset="UTF-8">
13     <meta http-equiv="X-UA-Compatible" content="IE=edge">
14     <meta name="viewport" content="width=device-width, initial-scale=1.0">
15     <link rel="stylesheet" href="../css/main.css">
16     <title>Login</title>
17 </head>
18
19 <body>
20     <h1>Connexion</h1>
21     <form method="post" action="">
22         <p>Pseudo : </p>
23         <input type="text" name="nickname" placeholder="pseudo du joueur" required>
24         <p>Mot de passe:</p>
25         <input type="password" name="password" placeholder="mot de passe" required>
26         <input type="submit" value="valider" name="btn">
27         <a href="./inscriptions.php">
28             <p>Je n'ai pas de compte!</p>
29         </a>
30         <p class="message"><?= $message ?></p>
31     </form>
32 </body>
33
34 </html>
```

## pages\rules.php

```

1 <?php
2
3 /**
4  * Description: Page des règles
5 */
6 require_once('../model/functions/user.php');
7 // Vérifie si l'utilisateur est connecté
8 if (!isset($_SESSION['nickname'])) {
9     header("Location: ./login.php");
10}
11?>
12<!DOCTYPE html>
13<html lang="fr">
14
15<head>
16    <meta charset="UTF-8">
17    <meta http-equiv="X-UA-Compatible" content="IE=edge">
18    <meta name="viewport" content="width=device-width, initial-scale=1.0">
19    <link rel="stylesheet" href="../css/main.css">
20    <title>Rules</title>
21
22<body class="game-background">
23    <section class="container">
24        <a href="../pages/account.php" class="link">
25            <p class="title">Pseudo: <?= $_SESSION["nickname"] ?></p>
26        </a>
27        <a href="../index.php" class="link"><button class="account">Retour <<< </button>
28    </a>
29        <ul>
30            <!--Règles-->
31            <li class="description">Slim Ninja est un jeu à but pédagogique qui permet aux
joueurs
32                d'apprendre les différents groupes d'aliments.
33            </li>
34            <li class="description">
35                Les joueurs peuvent choisir entre 5 modes de jeu (Céréale, Fruit, Légume,
Légumineuse, Féculent).
36            </li>
37            <li class="description">
38                Les différents aliments de toutes les catégories tombent aléatoirement sur
l'écran de haut vers le bas.
39            </li>
40            <li class="description">
41                Pour gagner des points les joueurs doivent appuyer avec le cursor de la
souris sur les bons aliments de la catégorie choisie.
42            </li>
43            <li class="description">
44                Si le joueur n'appuie pas sur le bon aliment de la catégorie avant qu'il
attegne le bas de l'écran il perd 2 points.
45            </li>
46            <li class="description">
47                Les joueurs perdent 2 points aussi lorsqu'ils appuient sur les mauvais
aliments (malbouffe).
48            </li>
49            <li class="description">
50                Des aliments malsains apparaissent de tant en temps sur l'écran,
si le joueur appuie sur l'un de ces aliments le jeu s'arrête immédiatement.
51            </li>

```

```
52     <li class="description">
53         Le jeu est aussi arrêté si le score du joueur devient négatif.
54     </li>
55     <li class="description">
56         En mode multijoueur le joueur qui a le score le plus élevé fait à défiler
les aliments de l'autre joueur plus vite.
57     </li>
58     <li class="description">
59         Le dernier joueur restant de la partie est le gagnant!
60     </li>
61 </ul>
62 <h2 class="description-h2">
63     Voici un schéma qui montre de quel groupe font partie les aliments présent dans
le jeu:
64 </h2>
65     
66 </section>
67 </body>
68
69 </html>
```