Slim Ninja

Documentation technique

Stefan Nikolic - 2023

Table des matières

[1.Table de versions 3](#_Toc135224023)

[2.Introduction 3](#_Toc135224024)

[3.Rappel de l’énoncé 3](#_Toc135224025)

[3.1 Organisation 3](#_Toc135224026)

[3.2 Objectif 3](#_Toc135224027)

[3.3 Spécifications 4](#_Toc135224028)

[3.4 Matériel et logiciels à disposition 4](#_Toc135224029)

[3.5 Livrables 5](#_Toc135224030)

[4.Planification 5](#_Toc135224031)

[4.1 Diagramme de Gantt prévisionnel 5](#_Toc135224032)

[4.2 Diagramme de Gantt finale : 6](#_Toc135224033)

[5.Méthodologie en six étapes 6](#_Toc135224034)

[5.1 S’informer 6](#_Toc135224035)

[5.2 Planifier 6](#_Toc135224036)

[5.3 Décider 6](#_Toc135224037)

[5.4 Réaliser 7](#_Toc135224038)

[5.5 Contrôler 7](#_Toc135224039)

[5.6 Évaluer 7](#_Toc135224040)

[6.Implémentation 7](#_Toc135224041)

[6.1 Base de données 7](#_Toc135224042)

[6.1A Table game : 9](#_Toc135224043)

[6.1B Table user 10](#_Toc135224044)

[6.1C Table item 10](#_Toc135224045)

[6.2 Fonctions PHP 11](#_Toc135224046)

[6.2A Ficher user.php 11](#_Toc135224047)

[6.2B Fichier food.php 13](#_Toc135224048)

[6.2C Ficher conf.php 16](#_Toc135224049)

[6.2D Fichier myDB.php 17](#_Toc135224050)

[6.3 API 17](#_Toc135224051)

[6.3A Fichier getFoods.php 18](#_Toc135224052)

[6.3B Fichier getUsers.php 18](#_Toc135224053)

[6.4 Fonctions Javascript 19](#_Toc135224054)

[6.4A Fichier user.js 19](#_Toc135224055)

[6.4B Fichier game.js 19](#_Toc135224056)

[6.4C Fichier UpdateUsers.js 25](#_Toc135224057)

[6.4D Fichier addBadFood.js 25](#_Toc135224058)

[6.4E Fichier getFoods.js 26](#_Toc135224059)

[6.4F Fichier sendDataToPhp.js 26](#_Toc135224060)

[7.Problèmes rencontrés 27](#_Toc135224061)

[8.Maquettes 27](#_Toc135224062)

[8.1 Page Inscription 28](#_Toc135224063)

[8.2 Page Login 28](#_Toc135224064)

[8.3 Page Index 29](#_Toc135224065)

[8.4 Page Account 29](#_Toc135224066)

[8.5 Page Rules 30](#_Toc135224067)

[8.6 Page Games 30](#_Toc135224068)

[8.7 Page Game 31](#_Toc135224069)

[8.8 Fin d’une partie 31](#_Toc135224070)

[8.9 Schéma des groupes d’aliments 32](#_Toc135224071)

[8.10 Schéma de navigation du site 32](#_Toc135224072)

[9.Plan de tests et rapport de tests 33](#_Toc135224073)

[9.1 Configuration : 33](#_Toc135224074)

[9.2 Tests : 33](#_Toc135224075)

[9.2A Fonctions PHP 33](#_Toc135224076)

[9.2B Fonctions JavaScript 39](#_Toc135224077)

[10.Conclusion 45](#_Toc135224078)

[10.1 Futurs améliorations 45](#_Toc135224079)

[10.2 Bilan personnel 45](#_Toc135224080)

[11.Glossaire 46](#_Toc135224081)

[12.Bibliographie 46](#_Toc135224082)

[12.1 ChatGPT 47](#_Toc135224083)

[13.Remerciement 47](#_Toc135224084)

[14.Code sources 48](#_Toc135224085)

[Annexe 1 48](#_Toc135224086)

# 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 appendre 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 choisis 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 fais 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>

## 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 diminue 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 détourer les images
* **World** 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 maitre 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

Une image contenant texte, ligne, nombre, Tracé

Description générée automatiquementVoici 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 digramme de Gantt prévisionnel

## 4.2 Diagramme de Gantt finale :

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

Figure 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 :**

Une image contenant texte, diagramme, capture d’écran, Parallèle

Description générée automatiquement

Figure 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 :**

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

Figure 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 (céréale, féculent, fruit, légume, légumineuse)**

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Figure 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 mis 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 fini 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 « idtem » 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 ()

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 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 ()

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 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()

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 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 ()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure Fonction LogOut()

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

#### Fonction DeleteUser ()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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 ()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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()

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure Fonction CreateGame()

Cette fonction permet de crée 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 à crée 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()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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.

#### Une image contenant texte, capture d’écran, Police, logiciel Description générée automatiquementLa fonction GetOnGoingGames ()

Figure Fonction GetOnGoingGames()

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

#### Une image contenant texte, capture d’écran, Police Description générée automatiquementLa fonction isUserInGame ()

Figure Fonction isUserInGame()

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

#### Une image contenant texte, Police, capture d’écran Description générée automatiquementLa fonction nbPlayersInGame ()

Figure 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()

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 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 crée.

#### Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran Description générée automatiquementLa fonction GetAllUserInformations ()

Figure 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

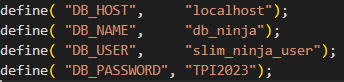


Figure Fichier conf.php

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

### 6.2D Fichier myDB.php

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure myDB.php

Dans ce fichier on va tout d’abord récupérer les donnés 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 (**A**pplication **P**rogram **I**nterface). 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 :

Une image contenant ligne, texte, capture d’écran, Police

Description générée automatiquement

Figure Fonctionnement de l'API

### 6.3A Fichier getFoods.php

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 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 ()

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 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 ()

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 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()

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure Fonction CreateFood()

La fonction suivante permet de crée 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 récupère depuis la fonction **RandomFoodFromList().**

#### Fonction draw()

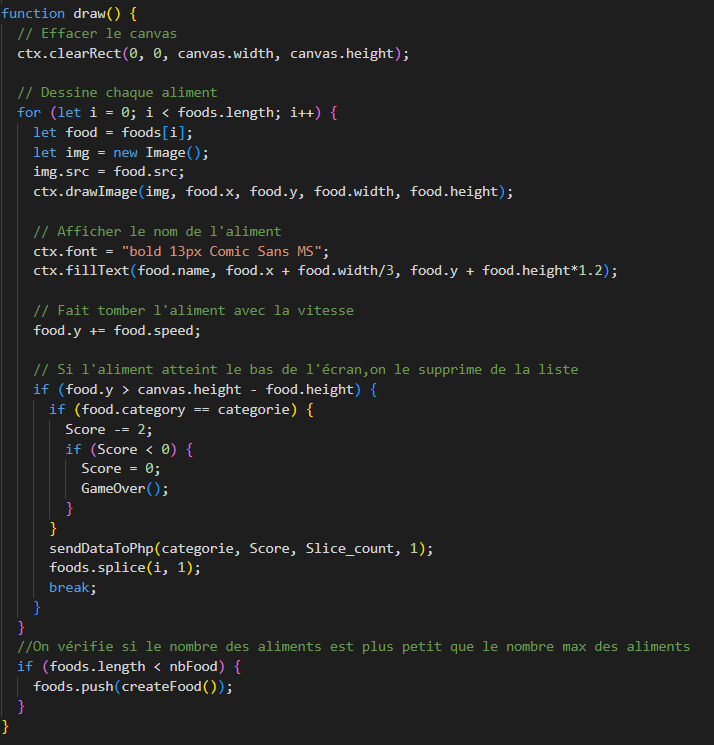


Figure Fonction draw()

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 fais 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.



Figure setInterval()

#### Fonction isPointerInsideFood ()

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Figure 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()

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Figure handleClickEvent()



Figure 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()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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 ()

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Figure Fonction resizeCanvas()



Figure 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 ()

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 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**

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure Évènement beforeunload

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

### 6.4C Fichier UpdateUsers.js

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 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 écoule 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 heurs puis l’afficher. Cette fonction est finalement exportée vers le fichier **game.js.**

### 6.4D Fichier addBadFood.js

#### Fonction addBadFood ()

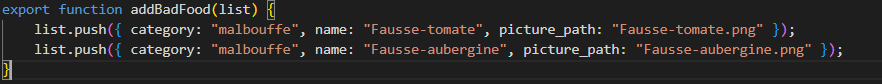


Figure Fonction addBadFood()

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()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure Fonction SendDataToPhp()

Cette fonction va envoyer des donnés 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.**

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 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 donné de la partie dans la base de donné 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 fonctionnalité 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



Figure Page Inscription

## 8.2 Page Login



Figure Page Login

## 8.3 Page Index

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure Page Login

## 8.4 Page Account

Une image contenant dessin humoristique, électroménager

Description générée automatiquement

Figure Page Account

## 8.5 Page Rules

Une image contenant texte, capture d’écran, électroménager

Description générée automatiquement

Figure Page Rules

## 8.6 Page Games

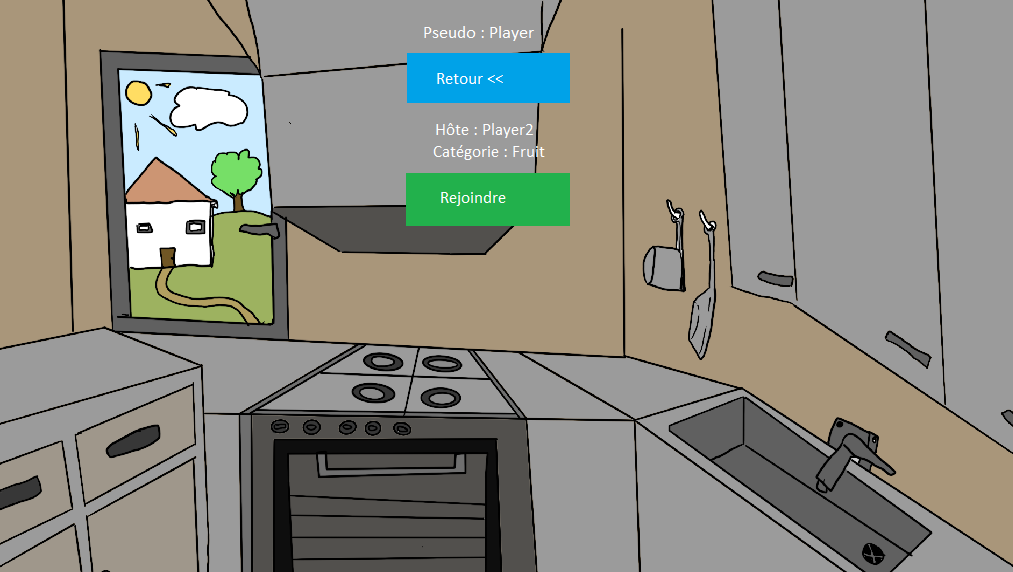


Figure Page Games

## 8.7 Page Game

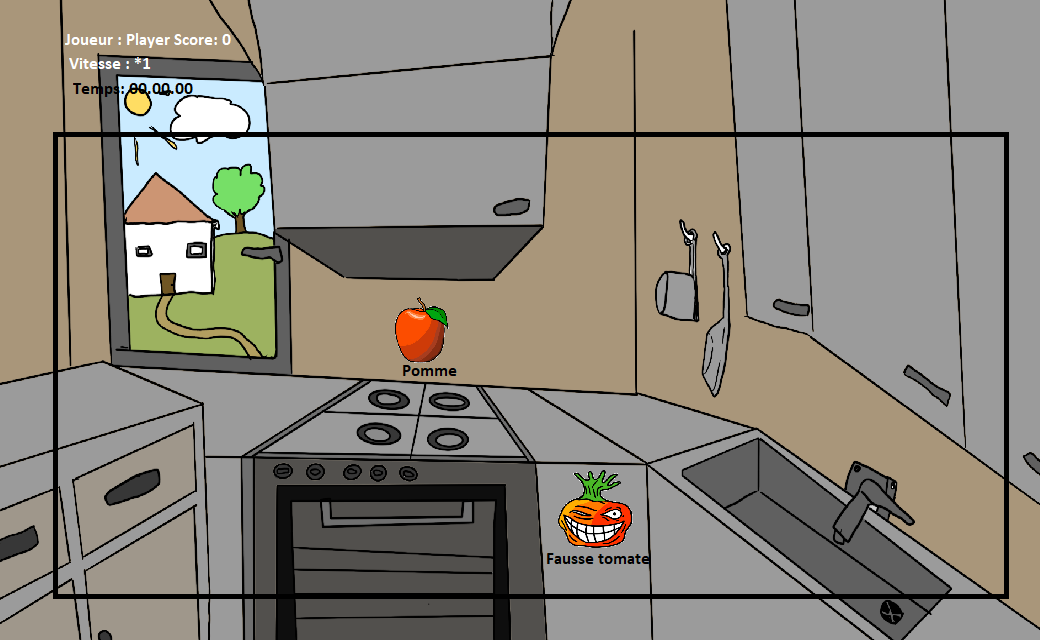


Figure Page Game

## 8.8 Fin d’une partie

Une image contenant dessin humoristique, électroménager

Description générée automatiquement

Figure La fin du jeu

## 8.9 Schéma des groupes d’aliments

Une image contenant texte, fruit, capture d’écran, légume

Description générée automatiquement

Figure Schéma des catégories d'aliments

## 8.10 Schéma de navigation du site

Une image contenant texte, diagramme, dessin, croquis

Description générée automatiquement

Figure 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 donnés que dans la deuxième étape.  Appuyer sur le bouton valider | Le message d’erreur s’affiche bien | Ok |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 crée 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 pore | 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 lorsque le score du joueur devient négatif | Lancer un jeu en mode fruit,  attendre qu’une pomme ou poire touche le bas de l’écran ou appuyer sur un aliment d’une autre catégorie | Le menu de fin de jeu est affiché | Ok |  |
| 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,  Joueur 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 motiver 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 2ème 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 écoule depuis le début de la partie J’ai utilisé le ChatGPT. Voici la conversion que je eu avec lui :

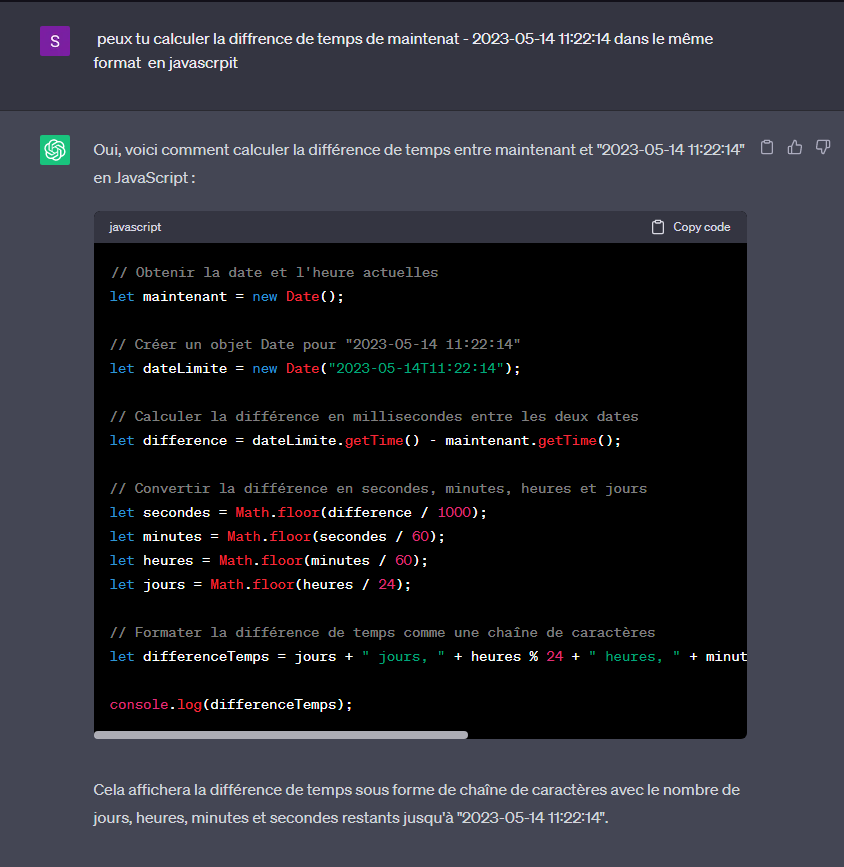


Figure Conversation ChatGPT

# 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