Slim Ninja

Documentation technique

Stefan Nikolic - 2023

Table des matières

[1.Table de versions 2](#_Toc133931158)

[2.Introduction 2](#_Toc133931159)

[3.Rappel de l’énoncé 2](#_Toc133931160)

[3.1 Organisation 2](#_Toc133931161)

[3.2 Objectif 2](#_Toc133931162)

[3.3 Spécifications 2](#_Toc133931163)

[3.4 Matériel et logiciels à disposition 3](#_Toc133931164)

[3.5 Livrables 4](#_Toc133931165)

[4.Planification 4](#_Toc133931166)

[4.1 Diagramme de Gantt prévisionnel 4](#_Toc133931167)

[5.Méthodologie 4](#_Toc133931168)

[6.Implémentation 5](#_Toc133931169)

[7.Problèmes rencontrés 5](#_Toc133931170)

[8.Maquettes 5](#_Toc133931171)

[9.Plan de tests et rapport de tests 5](#_Toc133931172)

[9.1 Configuration : 5](#_Toc133931173)

[9.2 Tests : 5](#_Toc133931174)

[10.Conclusion 6](#_Toc133931175)

[11.Biliographie 6](#_Toc133931176)

[12.Code sources 6](#_Toc133931177)

# 1.Table de versions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Nom | Date du 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 imposer 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 joueur(s) d’apprendre les différents groupes des aliments tout en jouant. Les joueurs doivent appuyer sur les bons aliments de la catégorie choisi pour gagner des points. Les aliments tombent du haut vers le bas aléatoirement sur l’écran. Si un joueur appui sur un fruit malsain ou si son score est négatif il perd la partie. Le dernier joueur restant gagne la partie gagne.

# 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 à l’utilisateur(s) d’apprendre les groupes d’aliments de manière ludique. Le jeu peut être jouer à deux. Le score de(s) l’utilisateur(s) augmente lorsque l’utilisateur appuie sur le bon aliment et diminue s’il appuie sur le(s) aliments qui ne correspondent pas à la catégorie choisie. Les fruits tombent verticalement de manière aléatoire. Si un joueur appui sur un fruit malsain ou si son score est négatif il perd la partie. Le dernier joueur restant gagne la partie gagne.

## 3.3 Spécifications

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

* Le jeu est en 2 dimensions
* Le joueur peut choisir parmi 5 catégories de jeu (fruit, légumes, légumineuses, céréales et féculant).
* Les aliments tombent verticalement de manière aléatoire. Leur positionnement horizontal est aussi aléatoire.
* Le nombres 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 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 rejoint une partie en cours sa vitesse du défilement est initialement la même que celle du joueur en face.
* 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 afficher 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 physique.

**Pour faire les spirites du jeu j’utilise le site web suivant :**

* <https://www.pixilart.com/>

**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
* **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 de 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)

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

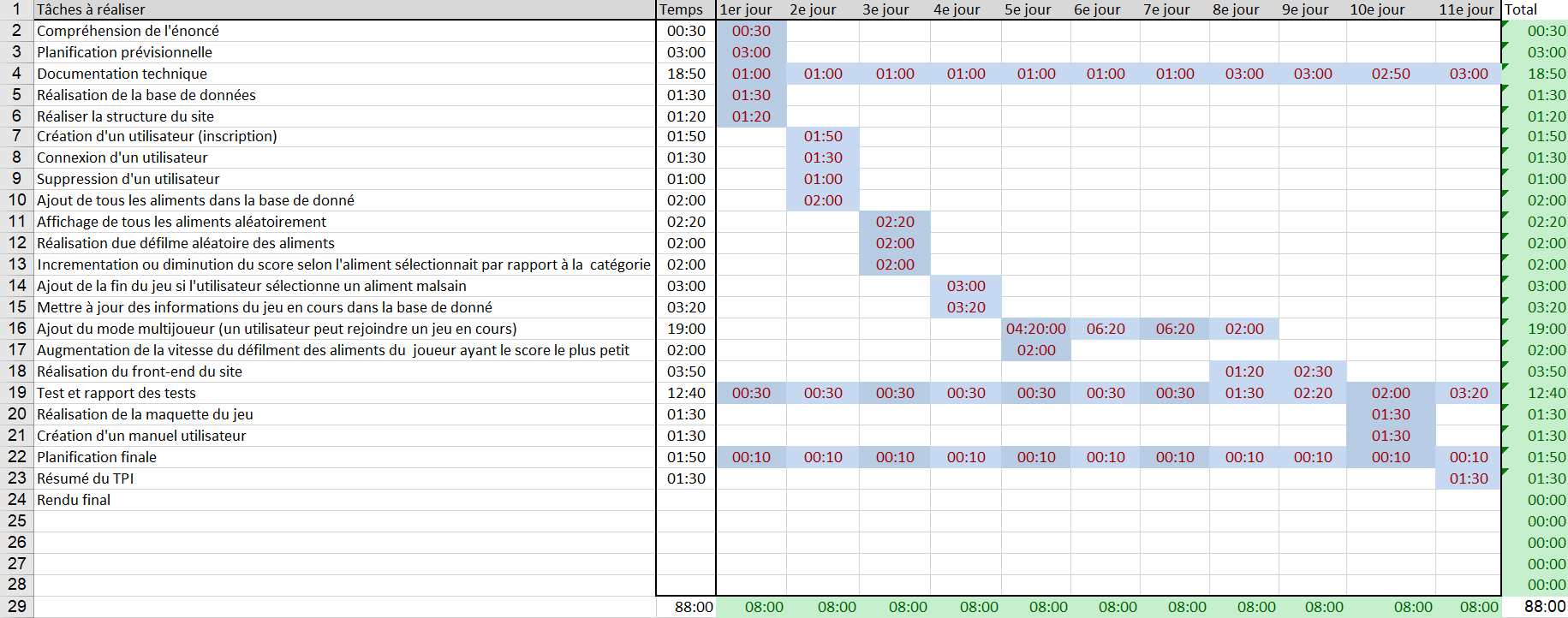
* Le carnet de bord
* Le code source

Le projet est disponible à l’adresse suivants :

<https://github.com/stefan1217/Nikolic-Stefan-M152>

# 4.Planification

## 4.1 Diagramme de Gantt prévisionnel

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

# 5.Méthodologie en six étapes

## 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 tous 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 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 choisi 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

# 

Dans l’énoncé de mon TPI le model conceptuel de la base de données à été mis à jour dans le document. J’ai décidé de changer le model conceptuel après en consultation avec mon prof de TPI pour des raisons techniques.

# 7.Problèmes rencontrés

# 8.Maquettes

# 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 : 4.9.7

## 9.2 Tests :

**1.Functions php**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

Contenu du test :

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

Contenu du test :

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

Contenu du test :

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

Contenu du test :

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

Contenu du test :

# 10.Conclusion

# 11.Biliographie

# 12.Code sources