[Date]

CHARDONNET Romain (qualité)

EL MANSOURI Redouane (chef)

HURDEBOURCQ Jeremy

ISAAC Elie

groupe 9

CESU

Serious game

Table des matières

[Introduction 2](#_Toc40726949)

[I) Plan 2](#_Toc40726950)

[a) Extraction 2](#_Toc40726951)

[b) Triage 6](#_Toc40726952)

[c) Evacuation 6](#_Toc40726953)

[II) Outils 7](#_Toc40726954)

[a) MVC 7](#_Toc40726955)

[b) Login 7](#_Toc40726956)

[c) Chat 7](#_Toc40726957)

[d) Diagramme de classes 7](#_Toc40726958)

[e) Base de données 7](#_Toc40726959)

[III) Gestion de projet 7](#_Toc40726960)

[Conclusion 7](#_Toc40726961)

# Introduction

Le projet CESU est un serious game de simulation d’urgence dans le but de contribuer aux formations du CHU de Dijon. Le jeu permet donc la connexion de plusieurs joueurs à l’aide d’un login et de communiquer à l’aide d’un chat afin de simuler le fonctionnement d’une situation d’urgence en équipe. L’application est constituée de trois grandes parties principales :

* L’extraction des victimes : c’est dans cette partie que le jeu commence, le joueur place les véhicules d’urgence et procède à l’extraction des victimes par les pompiers pour les déposer dans le PMA (poste médical avancé).
* Triage : il a lieu au PMA, les médecins du Samu procèdent au triage des victimes en fonction de leur cas respectif.
* Evacuation : cette partie est liée à l’évacuation des victimes dans les ambulances, on y trouve les ressources disponibles (nombre de places,…).

Le projet a été développé sous XAMPP pour la simulation en local lors du développement personnel. Ce dernier a été judicieux puisqu’en plus de faire fonctionner Apache, il intègre PhpMyAdmin. Nous avons donc pu aisément travailler avec une base de données local. Le projet dans sa globalité est hébergé sous le serveur Aragon de l’IEM.

Le projet a été développé avec les outils suivants : HTML, CSS, JQuery, Php et SQL (pour les requêtes gérées avec Php).

# Plan

## Extraction

La « première » partie de l’application est l’extraction. Elle possède un genre de « gameplay séquentiel » en quelque sorte puisque plusieurs étapes se succèdent dans cette partie. Lorsque l’on arrive sur cette page, on remarque une carte représentant une ville avec un incendie.

La première étape consiste à placer les véhicules d’urgence de manière stratégique aux alentours de l’incident. Les instructions sont données, un clic suivi des touches fléchées pour déplacer les véhicules puis espace pour immobiliser le véhicule. En tout, il y a deux véhicules de police et deux véhicules des pompiers à garer. On suppose pour cette partie que le Samu est déjà sur place et a installé le PMA. Celui-ci est donc fixé dès le début.

Une image contenant capture d’écran

Description générée automatiquement

Lors de l’immobilisation d’un véhicule des pompiers, son pompier respectif en sort pour procéder à l’évacuation des victimes.

Une image contenant capture d’écran, nombreux, botte

Description générée automatiquement

Ensuite, les pompiers doivent procéder à l’extraction des victimes. Lorsqu’un pompier rentre dans la zone de danger, son contour devient jaune, on peut donc extraire une victime.

Une image contenant jouet, très coloré, beaucoup, camion

Description générée automatiquement

Lorsque le pompier a récupéré une victime, son contour devient rouge, et le nombre de victimes est décrémenté au-dessus.

Une image contenant jouet, beaucoup, très coloré, horloge

Description générée automatiquement

On déplace le pompier dans la zone du PMA. Son contour devient bleu lorsque l’on peut déposer la victime.

Une image contenant jouet, camion, photo, anniversaire

Description générée automatiquement

Lors de l’appui sur « espace » à nouveau, le contour redevient blanc, et la victime à bien été déposée dans le PMA. C’est-à-dire qu’on a fait une insertion dans la base de données. Pour ce faire, on utilise un appel Ajax dans le fichier .js correspondant.

Le fichier vue1.js traite les fonctionnalités suivantes :

* Utilise les instance javascript des personnages et véhicules concernés
* Sélection des éléments cliqués en mettant le contour en blanc
* Déplacement de l’élément sélectionné avec les touches fléchées à l’aide d’une fonction move
* Adaptation de la couleur du contour à l’aide d’une fonction checkZone
* Garer un véhicule avec une méthode garer
* Porter une victime à l’aide d’une méthode porteVictime
* Déposer une victime dans le PMA à l’aide d’une méthode deposerVictime
* Génération de noms et nombres aléatoires à l’aide de deux fonctions respectives

Nous détaillerons par la suite la partie de dépôt d’une victime dans le PMA puisque c’est la partie qui va interagir avec le reste du projet.

Une image contenant assis, téléphone

Description générée automatiquement

Cette fonction est appelée à chaque dépôt de victimes. On y trouve à l’intérieur un appel ajax vers le fichier qui appelle le contrôleur correspondant.

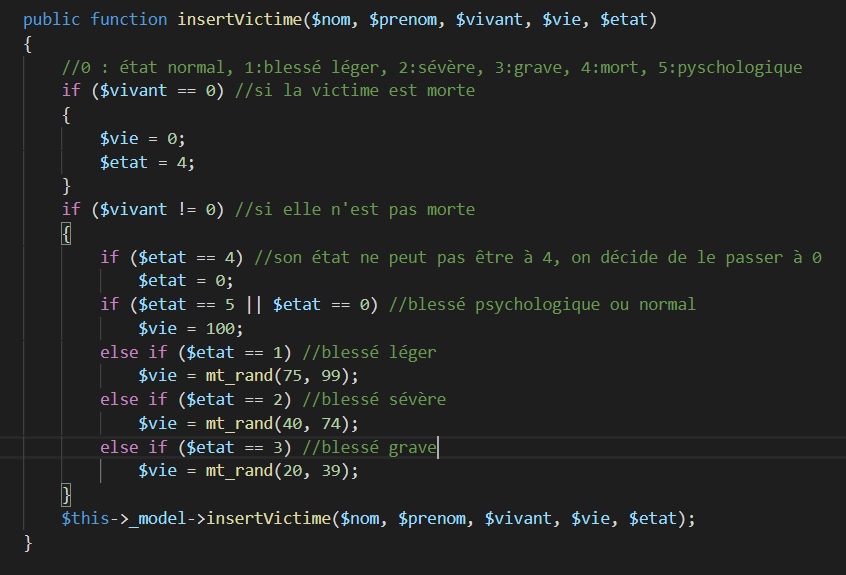
On utilise ici le type « post » puisque c’est la convention lors d’un enregistrement dans le serveur. Dans le cas d’une récupération de données, on aurait utilisé « get » mais dans les deux cas le système fonctionne.

Les données qu’on envoie dans « data » correspondent à chaque colonne de la table Victime présente dans la base de données. Pour les noms et les nombres, on utilise des fonctions de génération aléatoires.

Le nom de la fonction « insert » est également défini pour pouvoir l’identifier dans le fichier d’appel du contrôleur que voici :

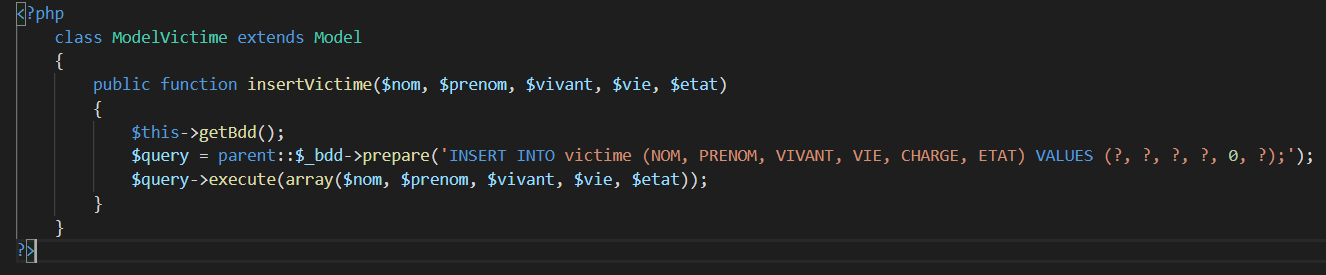


Regardons maintenant la méthode du contrôleur correspondante :



La particularité d’une victime est son état, on ajuste donc les plages de vie en fonction de ce dernier dans le contrôleur.

Enfin, l’insertion dans la base de données avec la fonction du modèle :



## Triage

## Evacuation

# Outils

## MVC

## Login

## Chat

## Diagramme de classes

## Base de données

# Gestion de projet

# Conclusion