# Escape Game

par RAN-dom Access Heroes

12/12/2024

# Introduction

Cette activité s'adresse à un groupe de trois à quatre personnes sans connaissance particulière en informatique qui disposent d'au moins un téléphone lisant des QR codes.

# Prérequis

Avant installation du matériel, il faut un espace clos de la taille d'un bureau, une connexion internet et d'un animateur.

# Histoire

Le site du centre de formation MNS est attaqué par un virus.

# Animateur

## Son rôle

Un recruteur qui met à l'épreuve les participants.

## Son objectif

devra aider les participants s' ils se trouvent bloqué sur un énigme mais également s'assurer l'intégrité du matériel.

# Matériels

Un ordinateur.
Un post-it.
Un classeur avec des documents :

- Procédure d'urgence.
- Tableau de conversion binaire - hexadécimal
- Explication code césar.

Une affiche (QRcode).
(Intercalaires dont un rouge).
Un dessous de verre (QRcode).
Un coffre avec double fond + matériel Arduino si nécessaire.
Un cordon USB type A mâle vers type B mâle.
Une clé USB contenant un fichier zippé protégé par un code.
Une mallette avec un code à 3 chiffres.

# A - Les fichiers du projet :

https://github.com/BrasseurTristan/Hackathon_MNS_2024

(La vidéo de la page d'accueil se trouve dans la racine du dossier Visual Studio)

# B - Site web :

## Les pages :

**- page "accueil" :**

départ de l'escape game, *statique*

**description** : on débute sur la page du site de la MNS. Nous avons refait cette page en utilisant une vidéo de la page déjà existante puis nous avons reproduit le footer. Sur ce footer deux liens sont cliquables : formations et administration. Le lien formation envoi vers la page vérolée tandis que le lien administration envoie sur la page login qui permet d'accéder au tableau de bord du site

**- page "admin login" :**

lien interne, dynamique.

**description** : page avec deux champs de saisie, identifiant et mot de passe. Le lien mot de passe oublié ouvre une pop-up qui donne un indice permettant de trouver le lieu où se trouve le mot de passe utile à la connexion administrateur

**- page "admin" :**

lien interne, statique

**description** : page avec une image de dashboard afin d'avoir un décor pour y glisser un message en code césar disant d'assembler les QR codes pour aller vers la page finale.

**- 3 pages indices :**

lien externe depuis les QR codes, statique

**description** : page avec un message indice pour aider les joueurs

**- page "virus" :**

lien externe, dynamique.

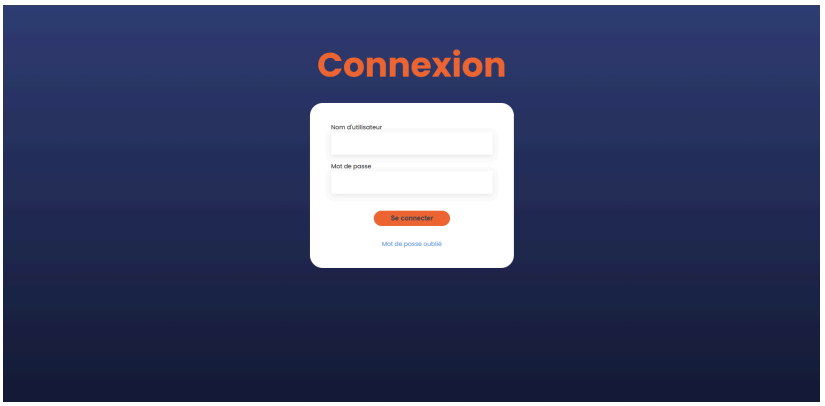
**description** : page avec un bouton pour arrêter le virus et afficher un message de victoire

**- page "jeu virus" :** page codée en JavaScript par Steve, à distance car souci de santé, mais bien présent pour aider son équipe. **Merci Steve !** C'est le petit jeu pour terminer cet escape game en beauté avec uniquement des commandes JavaScript

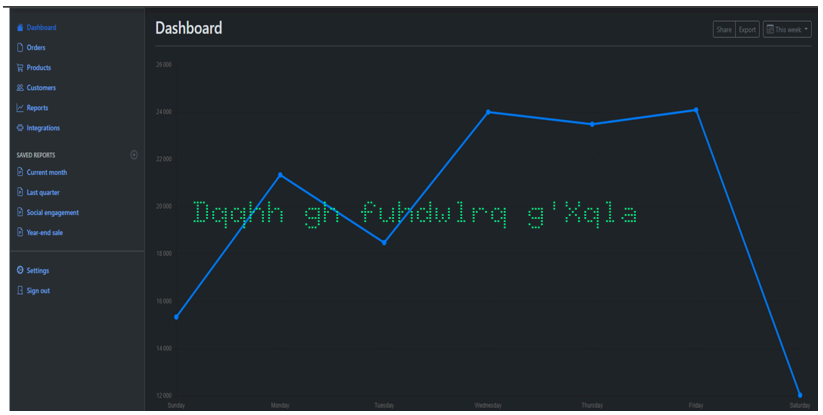
## Vue de la page d'accueil :

## Vue de le page hackée :

## Vue de la page login :

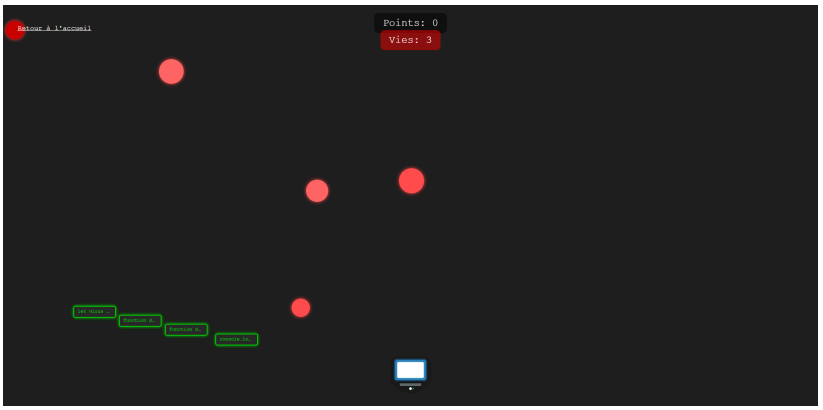



## Vue de la page admin :



**Vue de la page victoire :**

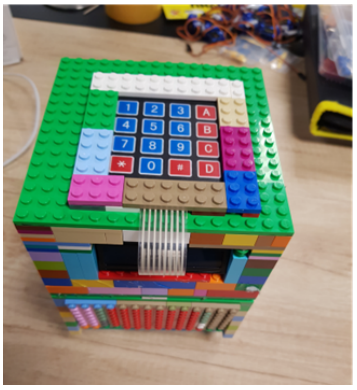
**Vue de la page jeu du virus :**




# C - Le coffret à digicode et double fond (prototype)

## 1) Description

Pendant la partie, les joueurs doivent ouvrir le coffre-fort interactif pour y trouver plusieurs éléments.



A son état de prototype, le coffre est une maquette effectuée en Lego, permettant d'imaginer une structure en bois ou autre matériaux adapté.

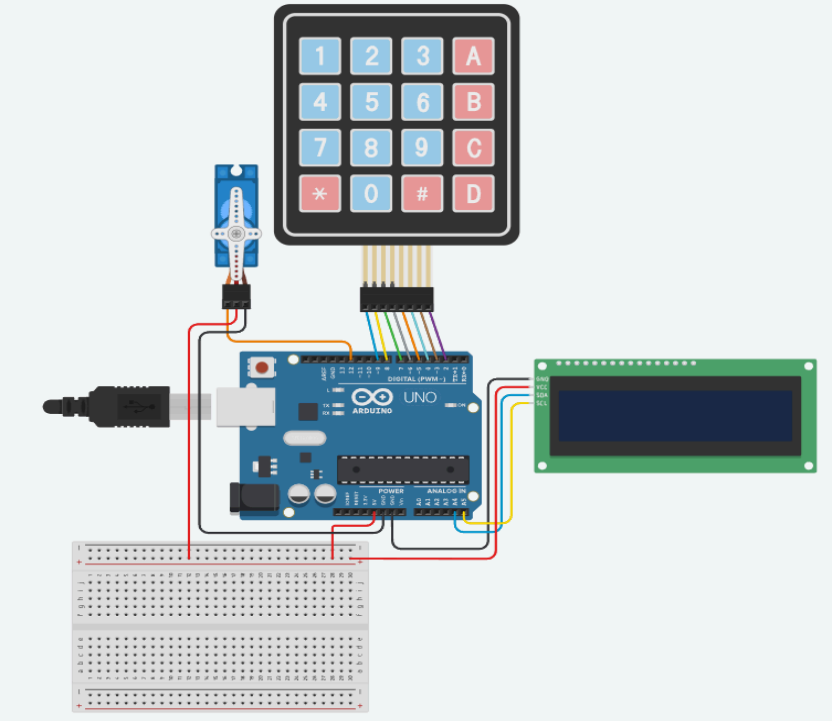
Une carte **Arduino Mega,** ou **UNO** permet d'automatiser certaines interactions avec le coffre.

L'ouverture de la porte du coffre se fait à l'aide d'un **« Keypad »** couplé avec un **afficheur QAPASS.**

Le couvercle de la boîte est actionné par un **servo-moteur SG90**, l'ouverture ainsi créée permet au joueur de savoir que le coffre est ouvert, en plus du message sur l'afficheur.

La boite est également composée d’un faux-fond, permettant de cacher d’autres éléments.

## 2) Schéma de câblage :



## 3) Code source pour la carte UNO

```cpp
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // Bibliothèque pour contrôler un écran LCD via I2C
#include <Keypad.h>            // Bibliothèque pour la gestion du clavier matriciel
#include "Servo.h"             // Bibliothèque pour contrôler le servo-moteur

Servo servo;                   // Création d'un objet Servo pour contrôler un servo-moteur
#define Password_Length 8      // Définition de la longueur maximale du mot de passe

int signalPin = 12;            // Définition du numéro de la broche pour le signal de verrouillage

// Déclarations des variables pour la gestion du mot de passe
char Data[Password_Length];    // Tableau pour stocker le mot de passe saisi
char Master[Password_Length] = "000000#"; // Mot de passe maître prédéfini
byte data_count = 0;           // Compteur pour suivre le nombre de caractères saisis
bool Pass_is_good;             // Variable pour vérifier si le mot de passe est correct
char customKey;                // Variable pour stocker la touche saisie

// Définition des dimensions du clavier matriciel (4x4)
const byte ROWS = 4;           // Nombre de lignes
const byte COLS = 4;           // Nombre de colonnes

// Définition des touches du clavier matriciel
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Définition des broches connectées aux lignes et colonnes du clavier
byte rowPins[ROWS] = {9, 8, 7, 6};   // Broches des lignes
byte colPins[COLS] = {5, 4, 3, 2};   // Broches des colonnes

// Initialisation de l'objet Keypad avec la configuration du clavier
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

// Initialisation de l'écran LCD (adresse I2C 0x27, écran 16x2)
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.init();                // Initialisation de l'écran LCD
  lcd.backlight();           // Activation du rétroéclairage de l'écran
  pinMode(signalPin, OUTPUT); // Configuration de la broche signalPin comme sortie
  servo.attach(12);          // Association du servo-moteur à la broche 10

  servo.write(0);            // Initialisation du servo à 0 degré
  delay(1000);               // Pause pour s'assurer que le servo atteint la position initiale
}

void loop() {
  lcd.setCursor(0, 0);       // Positionnement du curseur au début de l'écran
  lcd.print("Mot de passe:"); // Affichage du texte "Mot de passe:"

  customKey = customKeypad.getKey(); // Lecture de la touche saisie
  if (customKey) {                   // Si une touche a été pressée
    Data[data_count] = customKey;    // Stockage de la touche dans le tableau Data
    lcd.setCursor(data_count, 1);    // Positionnement du curseur pour afficher le caractère
    lcd.print(Data[data_count]);     // Affichage du caractère saisi
    data_count++;                    // Incrémentation du compteur de caractères
  }

  // Si le nombre de caractères saisis atteint la longueur du mot de passe
  if (data_count == Password_Length - 1) {
    lcd.clear();                     // Effacement de l'écran LCD

    // Comparaison du mot de passe saisi avec le mot de passe maître
    if (!strcmp(Data, Master)) {     // Si les deux mots de passe correspondent
      lcd.print("Ouvert");           // Affichage du message "Ouvert"
      digitalWrite(signalPin, HIGH); // Activation du signal pour ouvrir
      servo.write(120);               // Déplacement du servo-moteur à 90 degrés
      delay(5000);                   // Attente de 5 secondes
      digitalWrite(signalPin, LOW);  // Désactivation du signal
      servo.write(0);                // Retour du servo-moteur à sa position initiale
    } else {                         // Si le mot de passe est incorrect
      lcd.print("Incorrect");        // Affichage du message "Incorrect"
      delay(1000);                   // Attente d'une seconde
    }

    lcd.clear();                     // Effacement de l'écran LCD
    clearData();                     // Réinitialisation du tableau Data
  }
}

// Fonction pour réinitialiser les données saisies
void clearData() {
  while (data_count != 0) {          // Tant que le compteur n'est pas nul
    Data[data_count--] = 0;          // Réinitialisation des caractères un par un
  }
  return;
}
```

# D - Les QR codes

Vous pouvez utiliser des générateurs de QR codes en ligne gratuitement pour créer les QR code à l'adresse URL des pages quelles pointes

- Le dessous de verre (indice appliquer filtre rouge)

Indice : voir en rouge la solution

- Le poster (indice double fond dans le coffre)

**Je suis souvent caché,**
**Un secret bien gardé.**
**Dans une boîte ou un sac,**
**Je rends un objet moins flag.**
**Que suis-je ?**

- Papier dans le double fond du coffre (indice décryptage code césar)

Mon numéro préféré est 3

# F - Déroulement et listing des énigmes :

**Escape Game : SOS MNS**

**Introduction**

Dans cet Escape Game, les participants devront déjouer un virus informatique en suivant des énigmes et en explorant des indices cachés dans l'environnement. Le scénario allie manipulations physiques et recherches numériques pour immerger les joueurs dans une expérience captivante.

**Les étapes du jeu**

**1. Exploration de la page d'accueil**

La partie commence sur une page d'accueil comportant 4 onglets, mais seuls 2 sont fonctionnels : "formation" et "administration".

En cliquant sur "formation", une présentation des formations s'affiche brièvement avant qu'un virus ne s'ouvre dans un nouvel onglet. Cela perturbe les joueurs et les pousse à retourner sur la page d'accueil.

Le maître du jeu guide les participants en les encourageant à chercher des indices dans la pièce (QR codes) et à consulter une procédure d'urgence fournie dans une mallette.

**2. Trouver le lien vers la page login**

L'onglet "administration" mène à une page demandant un identifiant. Pour découvrir cet identifiant, les joueurs devront :

Scanner un QR code situé sur un poster, qui donne un indice sur l'existence du double fond du coffre Arduino.

Scanner un autre QR code caché sous un dessous de verre, qui fournit un indice menant au login de la page admin.

Utiliser une feuille rouge en plastique pour révéler certaines lettres sur un dessin dans la mallette. Cela permet de reconstituer l'identifiant : **"R0OTSUDO"**.

**3. Ouvrir le coffre Arduino**

Un post-it dissimulé sous l'ordinateur contient un code binaire à convertir en hexadécimal. Le tableau de conversion se trouve dans la mallette.

Une fois converti, ce code permet d'ouvrir le coffre Arduino à l'aide d'un clavier.

Le coffre contient une clé USB avec un fichier ZIP verrouillé par un mot de passe.

Le mot de passe est un nombre de 4 chiffres (3773), maquillé en lettres pour former le mot "ELLE". Un message dans la mallette indique : "ELLE connaît le code".

En déverrouillant le fichier ZIP, les joueurs trouvent un fichier .txt contenant le mot de passe admin : "P@ssw0rd!".

**4. Explorer le double fond du coffre**

En fouillant le coffre Arduino, les joueurs découvrent un double fond dissimulant un QR code.

Ce QR code mène à un indice permettant de comprendre la clé de déchiffrement César, nécessaire pour terminer le jeu.

**5. Déchiffrer l'énigme finale**

Avec les identifiants admin et le mot de passe, les joueurs accèdent au tableau de bord final sur la page admin.

Ils y trouvent un message crypté avec un chiffrement César +3 : "Dqqhh gh fuhdwlrq g'Xqla".

En appliquant le déchiffrement, ils obtiennent l'énigme : "Année de création d'Unix".

La réponse, 1969, permet de désactiver le virus sur sa page et de remporter la victoire.

**Bonus**

Pour prolonger l'expérience et initier les participants au codage, un petit jeu en JavaScript est proposé. Ce jeu permet d'apprendre les bases du langage de manière ludique.

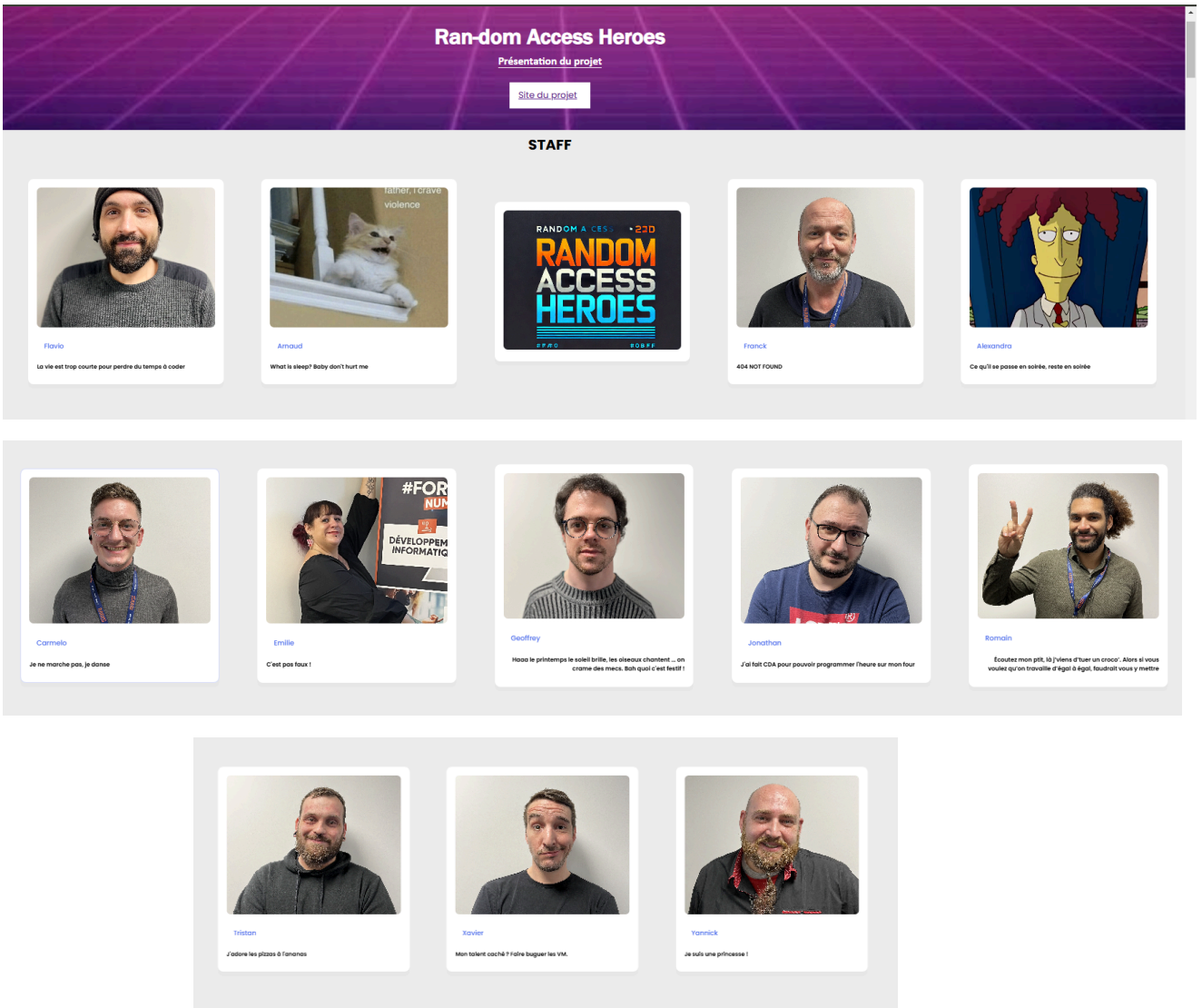
**Conclusion**

Cet Escape Game allie réflexion, exploration et apprentissage pour offrir une expérience à la fois immersive et pédagogique. Prêts à relever le défi ?

# G - Le serveur

Installation d’Apache, MariaDB et PHP sur un serveur Debian.

Création de la base de données et importation de fichier SQL contenu dans le projet (disponible sur le Github dans le dossier “db”).

(Pensez à changer le login et le mot de passe du fichier “config.php”).

# H - Site web de présentation :