Projet : jeu_combat
Cahier des charges :
JEU DE COMBAT
LE PRINCIPE
Il s'agit d'un jeu qui se joue à autant de joueurs que l'on veut (plus on est de fous, plus on rit !)
Le principe est simple :
on se créée et on paramètre un personnage (on lui donne un pseudo et un mot de passe pour pouvoir se
connecter à ce personnage et en prendre le contrôle)
il doit alors entrer sur le terrain de jeu, se battre ou se défendre, avancer dans toutes les pièces pour atteindre la
sortie
au cours des combats, on perd ou on gagne des points de vie. Si on les perd tous, le personnage meurt (fin de la
partie)
Le terrain de jeu est composé de 9 pièces successives, numérotée de 1 à 9 :
Entrée 1 2 3 4 5 6 7 8 9 Sortie
Les zones entrées et sorties sont neutres : on ne peut pas y combattre, ni y voir les autres personnages présents.
Le jeu comporte 3 pages
La création d'un personnage
La connexion (saisie du pseudo et du mot de passe d'un personnage) pour contrôler un personnage
Puis l'écran principal de jeu
Le choix d'un personnage consiste à paramétrer le personnage : son pseudo (unique), son mot de passe, et ses
caractéristiques (voir plus loin).
L'écran principal du jeu indique :
le rappel des caractéristiques du personnage
le numéro de la pièce où il se trouve
un bouton pour avancer dans la pièce suivante (si les caractéristiques le permettent)
un bouton pour revenir à la pièce précédente
de manière générale, des boutons pour les actions possibles
la liste des autres personnages se trouvant dans la pièce (cette liste est cliquable, il suffit de cliquer sur un
personnage pour l'attaquer). On ne voit pas les caractéristiques des autres joueurs, et on ne voit pas les autres
pièces.
l'historique des actions effectuées et subies (attaques effectuées, attaques subies, mouvements effectués), et
leur impact sur les 4 paramètres du personnage (voir ci-après : points de vie, force, agilité, résistance). Des
couleurs, éventuellement des pictos, permettent de visualiser d'un seul coup d'oeil les différents évènements, en

particulier les attaques subies.
Les informations qui changent sont mises à jour automatiquement toutes les demi-secondes.
1
CHOIX D'UN PERSONNAGE
Le choix d'un personnage consiste à :
choisir un pseudo (visible par les joueurs dans la même pièce) et un mot de passe
acquérir 100 points de vie automatiquement
répartir 15 points entre les 3 propriétés suivantes, avec minimum 3 points et maximum 10 points par propriété :
o force (force utilisée pour les attaques)
o agilité (pour les déplacements et les esquives)
o résistance (en défense)
A tout moment, en fonction de ses propriétés et des règles du jeu, le joueur à accès aux actions suivantes (donc les
actions ne sont visibles que si elles sont possibles)
avancer dans la pièce suivante
reculer
attaquer un personnage dans la même pièce (en cliquant sur le personnage)
transformer un point de force en point de résistance ou réciproquement.
TRANSFORMER UN POINT (FORCE OU RÉSISTANCE)
La transformation d'un point de force en point de résistance, ou réciproquement, consomme 3 points d'agilité.
Cela ne permet pas de dépasser 15 points de force ou 15 points de résistance.
DÉPLACEMENTS
Déplacement vers l'avant :
Un déplacement vers l'avant (dans la pièce suivante ou la zone de sortie) consomme des points d'agilité : il faut autant de
points d'agilité que le numéro de la pièce à atteindre, et 10 points pour passer dans la zone de sortie.
Si on n'a pas assez de points d'agilité, on n'a pas accès à la pièce suivante
Déplacement vers l'arrière :
Un déplacement vers l'arrière (dans la pièce précédente ou la zone d'entrée) est toujours possible et ne consomme pas
de points d'agilité. On peut le faire même avec zéro point d'agilité.
On gagne alors en points de vie le numéro de la pièce atteinte.
RESTER DANS UNE PIÈCE SANS RIEN FAIRE
Si on reste dans une pièce sans rien faire (sans attaquer) pendant 3 secondes, on récupère 1 point d'agilité.
Cela ne s'applique pas à la zone d'entrée.
Attention : les points d'agilité sont plafonnés à 15 !
2
ATTAQUER
Lorsqu'on clique sur un personnage (donc présent dans la même pièce), cela signifie qu'on l'attaque.

L'attaque est alors automatique, et se fait avec une force déterminée et que l'on ne peut pas choisir (cette force
d'attaque est utilisée pour déterminer le déroulement du combat, voir chapitre suivant) : la force de l'attaque est la force
de l'attaquant.
Si l'adversaire esquive et que l'on a 10 points de force ou plus, un point de force devient un point de résistance.
Si on gagne le combat, on récupère un point d'agilité (ça motive !), ou un point de vie si on a déjà 15 points d'agilité.
Si on gagne le combat et que en plus l'on tue l'adversaire, on récupère en plus les points de vie qui lui restaient juste
avant le combat.
Si on perd le combat : on perd 1 point de vie.

SUBIR UNE ATTAQUE

Lorsque l'on subit une attaque d'une certaine force, on se défend, voire on riposte.
On peut donc subir une attaque alors même que l'on n'est pas connecté.
Cette opération est automatique (et apparait dans l'historique de l'utilisateur).
Si notre agilité dépasse la force d'attaque d'au moins 3 points, on esquive. Personne n'a alors gagné ou perdu le combat,
et on perd 1 point d'agilité.
Si notre force est supérieure strictement à celle de l'attaque, on riposte : voir ci-après la riposte. On gagne le combat et
un point de vie si on gagne la riposte, on perd le combat et 2 points de vie si on perd la riposte.
Sinon, on se défend : si notre résistance est supérieure ou égale à la force de l'attaque, on gagne le combat, si elle est
inférieure, on le perd et on perd en points de vie la différence entre notre résistance et la force de l'attaque.

RIPOSTE

La riposte fonctionne comme une attaque, mais est déclenchée automatiquement.
On attaque alors notre attaquant avec toute notre force disponible, et notre attaquant applique les règles "subir une
attaque".

QUELQUES CONSEILS

Bien définir l'ergonomie pour que cela soit jouable (il faut bien sûr utiliser Ajax)
Je vous conseille de réaliser une méthode par action possible. Et donc par exemple, quand un personnage
attaque, on a un scénario du type :
◦ A attaque B avec la force N (il appelle la méthode avec ses points de force, donc A attaque avec sa force,
mais B est attaqué avec la force choisie par A)
◦ donc B subit une attaque de force N, et indique le résultat à A
◦ les objets A et B gérent chacun les propriétés de leur personnage respectif

CSS :
style.css

```css
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.flex{
    display: flex;
    flex-wrap: wrap;
}

.fs14{font-size: 14px;}
.fs12{
    font-size: 12px;
}
.fs20{
    font-size: 20px;
}
.fs16{
    font-size: 16px;
}
.fs10{
    font-size: 10px;
}
.fs8{
    font-size: 8px;
}

.wfit{
    width: fit-content;
}

.w100{
    width: 100%;
}

.w75p{
```

```
    width: 75px;
}
.w100p{
    width: 100px;
}
.w120p{
    width: 120px;
}
.w80p{width: 80px;}
.w130p{width: 130px;}
.w140p{
    width: 140px;
}
.w150p{
    width: 150px;
}
.w200p{
    width: 200px;
}
.w300p{
    width: 300px;
}
.w34p{width: 34px; height: 18px; align-self: center;}
.mrlauto{
    margin-left: auto;
    margin-right: auto;
}
body{
    position: relative;
}

.fond{
    position: fixed;
    top: 0;
    left: 0;
    z-index: -1;
    width: 100%;
    height: 100vh;
    background-size: cover;
    background-repeat: no-repeat;
```

```
    background-position: center;
}

.regle{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/regle.jpg");}
.accueil{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.4)), url("../assets/connexion.jpg");}
.creation{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.4)), url("../assets/creation.jpg");}
.entree{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.4)), url("../assets/0.jpg");}
.un{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/1.jpg");}
.deux{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/2.jpg");}
.trois{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/3.jpg");}
.quatre{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.4)), url("../assets/4.jpg");}
.cinq{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/5.jpg");}
.six{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/6.jpg");}
.sept{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/7.jpg");}
.huit{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/8.jpg");}
.neuf{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/9.jpg");}
.sortie{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.4)), url("../assets/10.jpg");}
.mort{background-image: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0,
0.4)), url("../assets/11.jpg");}

.w110p{width: 110px;}

li {list-style: none;}

.a-around{
    align-content: space-around;
```

```
}
.j-center{
    justify-content: center;
}
.j-between{
    justify-content: space-between;
}

.j-end{
    justify-content: flex-end;
}

.a-center{
    align-items: center;
}
.ml16{margin-left: 16px;}
.mt32{
    margin-top: 32px;
}

.mt80{
    margin-top: 80px;
}
.mt160{margin-top: 160px;}

.mt8{
    margin-top: 8px;
}

.mt16{
    margin-top: 16px;
}
.mb32{margin-bottom: 32px;}

.mr20{
    margin-right: 20px;
}
.mr60{
    margin-right: 60px;
}
```

```css
.gap8{
    gap: 8px;
}
.gap32{
    gap: 32px;
}

.mt200{
    margin-top: 200px;
}
.mt100{
    margin-top: 100px;
}

.txt-center{
    text-align: center;
}

.txt-end{
    text-align: end;
}

.txt-start{
    text-align: start;
}

.mt48{
    margin-top: 48px;
}


.yellow{ color: rgb(240, 240, 11);}
.red{
    color: rgb(255, 0, 0);
}
.green{color: rgb(0, 255, 0);}
.orange{color: rgb(255, 166, 0);}

.column{
```

```
    flex-direction: column;
}

button, input[type="submit"]{
    padding: 4px 8px;
}
.btnInactif{
    background-color: rgb(138, 138, 35);
}
.btnActif{
    background-color: rgb(240, 240, 11);
}

.hover:hover{
    cursor: pointer;
}

input[type="number"]{
    width: 50px;
}


.titre{
    height: 80px;
}

.container{
    height: calc(100vh - 80px);
    align-content: center;
}

.containerHisto{
    height: 140px;
    overflow-y: auto;
}

.listContainer{
    height: 100%;
}
```

```css
.containerAdv{
    height: 80px;
    overflow-y: auto;
}

.a-end{
    align-self: flex-end;
}
```

JS :
combat.js

```
let forToRes = document.getElementById("forToRes");
```

```js
let resToFor = document.getElementById("resToFor");
let force = document.getElementById("for");
let agi = document.getElementById("agi");
let res = document.getElementById("res");
let preced = document.getElementById("preced");
let suiv = document.getElementById("suiv");
let pv = document.getElementById("pv");
let position = document.getElementById("piece");
let adv = document.getElementById("adv");
let histo = document.getElementById("histo");
let fond = document.querySelector(".containFond");
let ctPv = document.getElementById("ctPv");
let ctAgi = document.getElementById("ctAgi");


// surveillance del'historique des action
function surveilleHisto(){
    fetch("surveiller_historique.php")
    .then(resp => {
        return resp.text();
    })
    .then( liste => {
        histo.innerHTML = liste;
    })
}
setInterval(surveilleHisto, 500);
```

```
// surveillance des adversaires présent dans la pièce
function surveilleAdv(){
    fetch("surveiller_adversaire.php")
    .then(resp => {
        return resp.text();
    })
    .then( liste => {
        detacheSurveilleAttaque();
        adv.innerHTML = liste;
        attacheSurveilleAttaque();
    })
}
setInterval(surveilleAdv, 500);


// surveille l'inactivité du joueur pour ajouter des points d'agilité
function surveilleInactif(){
    fetch("rester_inactif.php")
    .then(resp => {
        return resp.json();
    })
    .then( inactif => {
        afficheAgi(inactif);
    })
}
setInterval(surveilleInactif, 3000);


// surveille les changement des attributs du personnage et les affiche
toutes les demi seconde
function surveilleAttributs(){
    fetch("surveiller_caracteristiques.php")
    .then(resp => {
        return resp.json();
    })
    .then(data => {
        affichePv(data)
        afficheAttributs(data);
        afficheAgi(data);
```

```
        afficheBtn(data);
        detacheSurveilleClick();
        attacheSurveilleClick();
    })
}
setInterval(surveilleAttributs, 500);


function afficheBtn(data){
    // applique la class btn actif ou inactif selon les data reçu
    if(data.btnPreced == 1){
        if(preced.classList.contains("btnInactif")){
            preced.classList.remove("btnInactif");
        }
        preced.classList.add("btnActif")
    }
    else{
        if(preced.classList.contains("btnActif")){
            preced.classList.remove("btnActif");
        }
        preced.classList.add("btnInactif")
    }
    if(data.btnSuiv == 1){
        if(suiv.classList.contains("btnInactif")){
            suiv.classList.remove("btnInactif");
        }
        suiv.classList.add("btnActif")
    }
    else{
        if(suiv.classList.contains("btnActif")){
            suiv.classList.remove("btnActif");
        }
        suiv.classList.add("btnInactif")
    }
    if(data.btnForToRes == 1){
        if(forToRes.classList.contains("btnInactif")){
            forToRes.classList.remove("btnInactif");
        }
        forToRes.classList.add("btnActif")
    }
```

```
    else{
        if(forToRes.classList.contains("btnActif")){
            forToRes.classList.remove("btnActif");
        }
        forToRes.classList.add("btnInactif")
    }
    if(data.btnResToFor == 1){
        if(resToFor.classList.contains("btnInactif")){
            resToFor.classList.remove("btnInactif");
        }
        resToFor.classList.add("btnActif")
    }
    else{
        if(resToFor.classList.contains("btnActif")){
            resToFor.classList.remove("btnActif");
        }
        resToFor.classList.add("btnInactif")
    }
}

function detacheSurveilleClick(){
    forToRes.removeEventListener("click", clickForToRes);
    resToFor.removeEventListener("click", clickResToFor)
    suiv.removeEventListener("click", pieceSuivante)
    preced.removeEventListener("click", piecePrecedente)
}



function attacheSurveilleClick(){
    // surveille le click des boutons modif attributs et lance la fonction
associé
    if(forToRes.classList.contains("btnActif")){
        forToRes.addEventListener("click", clickForToRes)
    }
    if(resToFor.classList.contains("btnActif")){
        resToFor.addEventListener("click", clickResToFor)
    }
    // surveille le click des btn change piece et lance fonction associé
    if(suiv.classList.contains("btnActif")){
```

```
        suiv.addEventListener("click", pieceSuivante)
    }
    if(preced.classList.contains("btnActif")){
        preced.addEventListener("click", piecePrecedente)
    }
}

function detacheSurveilleAttaque(){
    let adversaires = adv.querySelectorAll("button");
    adversaires.forEach(adversaire =>{
        let id = adversaire.dataset.idadv;
        adversaire.removeEventListener("click", () => {
            attaqueAdv(id)
        })
    })
}



function attacheSurveilleAttaque(){
    let adversaires = adv.querySelectorAll("button");
    adversaires.forEach(adversaire =>{
        let id = adversaire.dataset.idadv;
        if(adversaire.classList.contains("btnActif")){
            adversaire.addEventListener("click", () => {
                attaqueAdv(id)
            })
        }
    })
}

function attaqueAdv(id){
    console.log(`attaque ${id}`);
    fetch(`attaquer.php?idAdv=${id}`)
    .then(resp => {
        return resp.text();
    })
    .then( liste => {
        console.log(liste);
    })
```

```
}

function clickForToRes(){
    fetch("forToRes.php")
    .then(resp => {
        return resp.json();
    })
    .then(data => {
        afficheAttributs(data);
    })
}
function clickResToFor(){
    fetch("resToFor.php")
    .then(resp => {
        return resp.json();
    })
    .then(data => {
        afficheAttributs(data);
    })
}
function pieceSuivante(){
    fetch("suivante.php")
    .then(resp => {
        return resp.json();
    })
    .then(data => {
        affichePiece(data);
        afficheAgi(data);
    })
}
function piecePrecedente(){
    fetch("precedente.php")
    .then(resp => {
        return resp.json();
    })
    .then(data => {
        affichePiece(data);
        affichePv(data);
    })
}
```

```
function afficheAttributs(data){
    // role : affiche les bon attributs
    // parametre : data - tableau d'attributs ("for" => $for)
    force.innerText = `⚔️ : ${data.for}/15`;
    res.innerText = `🛡️ : ${data.res}/15`;
}

function affichePiece(data){
    // role : affiche les bon attributs
    // parametre : data - tableau d'attributs ("for" => $for)
    ctPv.innerText = `+${data.preced} 🩸`;
    ctAgi.innerText = `-${data.suiv} ⚡`;
    afficheFond(data.piece);
}

function affichePv(data){
    // role : affiche les pv
    // parametre : data - tableau d'elements
    pv.innerText = `🩸 : ${data.pv}/100`;
    if(data.pv > 75){
        pv.classList.add("green");
    }else if(data.pv > 50 && pv <= 75){
        pv.classList.add("yellow");
    }else if(data.pv > 25 && pv <= 50){
        pv.classList.add("orange");
    }else{
        pv.classList.add("red");
    }if(data.pv == 0){
        fond.innerHTML = "";
        ctPv.innerText = "";
        ctAgi.innerText = "";
        preced.innerText = "⚜️ Salle des âmes perdues ⚜️";
        position.innerText = "⚜️ Salle des âmes perdues ⚜️";
        suiv.innerText = "⚜️ Salle des âmes perdues ⚜️";
        fond.innerHTML = `<div class="fond mort"></div>`;
    }
}
```

```
function afficheAgi(data){
    // role : affiche les point d'agi
    // parametre : data - tableau d'elements
    agi.innerText = `⚡ : ${data.agi}/15`;
}

function afficheFond(piece){
    fond.innerHTML = "";
    if(piece == 0){
        preced.innerText = "🏰 Entrée 🏰";
        position.innerText = "🏰 Entrée 🏰";
        suiv.innerText = "🌳 Forêt enchantée 🌳";
        fond.innerHTML = `<div class="fond entree"></div>`;
    }else if(piece == 1){
        preced.innerText = "🏰 Entrée 🏰";
        position.innerText = "🌳 Forêt enchantée 🌳";
        suiv.innerText = "❄️ Caverne de glace ❄️";
        fond.innerHTML = `<div class="fond un"></div>`;
    }else if(piece == 2){
        preced.innerText = "🌳 Forêt enchantée 🌳";
        position.innerText = "❄️ Caverne de glace ❄️";
        suiv.innerText = "🔥 Temple du feu 🔥";
        fond.innerHTML = `<div class="fond deux"></div>`;
    }else if(piece == 3){
        preced.innerText = "❄️ Caverne de glace ❄️";
        position.innerText = "🔥 Temple du feu 🔥";
        suiv.innerText = "📚 Bibliothèque abandonnée 📚";
        fond.innerHTML = `<div class="fond trois"></div>`;
    }else if(piece == 4){
        preced.innerText = "🔥 Temple du feu 🔥";
        position.innerText = "📚 Bibliothèque abandonnée 📚";
        suiv.innerText = "🌷 Jardin souterrain 🌷";
        fond.innerHTML = `<div class="fond quatre"></div>`;
    }else if(piece == 5){
        preced.innerText = "📚 Bibliothèque abandonnée 📚";
        position.innerText = "🌷 Jardin souterrain 🌷";
        suiv.innerText = "💰 Chambre des trésors 💰";
        fond.innerHTML = `<div class="fond cinq"></div>`;
    }else if(piece == 6){
```

```
        preced.innerText = "🌷 Jardin souterrain 🌷";
        position.innerText = "💰 Chambre des trésors 💰";
        suiv.innerText = "☠️ Marais toxique ☠️";
        fond.innerHTML = `<div class="fond six"></div>`;
    }else if(piece == 7){
        preced.innerText = "💰 Chambre des trésors 💰";
        position.innerText = "☠️ Marais toxique ☠️";
        suiv.innerText = "👻 Cimetière hanté 👻";
        fond.innerHTML = `<div class="fond sept"></div>`;
    }else if(piece == 8){
        preced.innerText = "☠️ Marais toxique ☠️";
        position.innerText = "👻 Cimetière hanté 👻";
        suiv.innerText = "👹 Salle du boss 👹";
        fond.innerHTML = `<div class="fond huit"></div>`;
    }else if(piece == 9){
        preced.innerText = "👻 Cimetière hanté 👻";
        position.innerText = "👹 Salle du boss 👹";
        suiv.innerText = "🚪 Sortie 🚪";
        fond.innerHTML = `<div class="fond neuf"></div>`;
    }else if(piece == 10){
        preced.innerText = "👹 Salle du boss 👹";
        position.innerText = "🚪 Sortie 🚪";
        suiv.innerText = "🚪 Sortie 🚪";
        fond.innerHTML = `<div class="fond sortie"></div>`;
    }
}
```

connexion.js

```
let btnMdpConnect = document.getElementById("btnMdpConnect");
let mdpConnect = document.getElementById("mdp_connexion");

btnMdpConnect.addEventListener("click", (e) => {
    if(mdpConnect.type === "password"){
        mdpConnect.type = "text";
    }else{
        mdpConnect.type = "password";
    }
})
```

Form_crea.js

```
let btnMdpCrea = document.getElementById("btnMdpCrea");
let mdpCrea = document.getElementById("mdp_creation");
let btnMdpVerif = document.getElementById("btnMdpVerif");
let mdpVerif = document.getElementById("mdpVerif_creation");

btnMdpCrea.addEventListener("click", (e) => {
    if(mdpCrea.type === "password"){
        mdpCrea.type = "text";
    }else{
        mdpCrea.type = "password";
    }
})

btnMdpVerif.addEventListener("click", (e) => {
    if(mdpVerif.type === "password"){
        mdpVerif.type = "text";
    }else{
        mdpVerif.type = "password";
    }
})
```

Utils

Init.php

```
<?php

/* code d'initialisation à insérer en début de chaque contrôleur */

// gestion et affichage des erreurs
ini_set("display_errors", 1);       // Aficher les erreurs
error_reporting(E_ALL);             // Toutes les erreurs


include "utils/model.php";

// ouverture de la base de donnée
$bdd = _model::bdd();

// propriété de debug
```

```php
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

// chargement des librairies
include "modeles/personnage.php";
include "modeles/historique.php";
include "utils/session.php";


// activation du mécanisme de session
session_activation();
```

Model.php

```php
<?php

/*
Classe _model : classe générique de gestion des objets du modèle de
données
(on a un _ dans le nom pour être sûr de ne pas avoir de table de ce nom)


Pour l'utiliser, on a mles méthode
    __construct(id) : instancie un nouvelle objet et le charge avec son id
    getTarget($fieldName) : retourner un objet pointé par un champ
    load(id) : chargement d'un objet depuis la BDD par son id
    is() : indique si l'objet est chargé / existe (true si existe, false
sinon)
    get(nomChamp) : récupération de la valeur d'un champ
    id() : récupère l'id
    set(nomChamp, valeur) : affectation d'une valeur à un champ
    insert() : ajout de l'objet courant dans la BDD
    update() : mise à jour de l'objet courant dans la BDD
    delete() : suppression de l'objet courant de la BDD
    listAll(+/-tri, +/-tri) : récupère tout les champs de tous les objet
de la table

*/
```

```php
class _model {

    // Attributs :
    // Description du modèle de l'objet (de la table)

    protected $table = "";       // Nom de la table, à valoriser pour les classes réelles;
    protected $fields = [];      // Liste des noms des champs, avec le type de champ
    // [ "nomChamp1" => typeChamp1, etc...]
    // Types de champs gérés : "NUM", "TXT", "LINK"
    protected $links = [];       // Liste des liens sortants :
        //tableau qui pour cahque lien met en index le nom du champ qui est un lien, et en valeur le nom de l'objet
        //  (exemple : [ "fournisseur" => "fournisseur"])


    // Stoker un objet précis
    protected $id = 0;       // id de l'objet chargé
    protected $values = []; // On stockera les valeurs sous la forme [ "nomChamp1" => valeur1, ... ]
    protected $targets = [];    // On stockera pour les liens [ "nomChamp" => objetLié, .. ]

    // Base de données ouverte
    protected static $bdd;

    // Ouverture de la base de données
    static function bdd() {
        // Rôle : retourne l'objet PDO vers la bdd, le créée si nécessaire
        // Paramètres néant
        // Retour : l'objet PDO

        if (empty(static::$bdd)) {
            static::$bdd = new PDO("mysql:host=localhost;dbname=projets_combat_alaugier;charset=UTF8", "alaugier", "9FPp96F9l?T");
        }
```

```php
        return static::$bdd;

    }


    // Constructeur

    function __construct($id = null) {
        // Cette fonction se déclenche à chaque fois que l'on instancie un objet (new nomClasse())
        // Les paramètres du constructeur devront être valorisés dans les parenthèse du new nomClasse()
        // rôle : charger l'objet correspondant à l'id (si non null)
        // paramètre : l'id de la ligne à chargé
        // retour : constructeur, pas de retour

        // Si l'id est non null
        //  Charger l'objet avec cet id
        if ( ! is_null($id)) {
            $this->load($id);
        }

    }

    // méthodes

    function is() {
        // Rôle : dire si l'objet est chargé (si il y a un contact de la BDD dedans)
        // Paramères : néant
        // Retour : true si il est chargé, false sinon

        return ! empty($this->id);
            // empty recouvre variable non initialisée, variable valant null, et toutes les valeurs apparentées à false (false, 0, "", [])


    }

    // Getters : récupérer les attributs
```

```
    // Au lieu de $contact->getEmail(), on va avoir une syntaxe
$contact->get("email")

    function get($fieldName) {
        // Rôle : récupérer la valeur d'un attribut
        // Paramètres :
        //      $fieldName : nom de l'attribut
        // Retour : la valeur de l'attribut (chaine vide si l'attribut
n'existe)

        // On a la valeur dans l'attribut values, à l'index qui a le même
nom que l'attribut cherché
        // l'attribut values est accessible $this->values
        // l'index qui nous intéresse est dans $fieldName

        //  cad $this->values[$fieldName];

        // On contrôle que la valeur existe, sinon, on retourne ""
        // Si la valeur existe (isset(....)) retourne la valeur, sinon
retourne ""
        if (isset($this->values[$fieldName])) {
            return $this->values[$fieldName];
        } else {
            return "";
        }

    }

    function getTarget($fieldName) {
        // Rôle : retourner un objet pointé par un champ
        // paramètre :
        //      $fieldName : nom du champ
        // Retour : objet (d'une classe héritée de la classe _model),
chargé avec l'objet pointé
        //       si on ne trouve pas :
        //          si champ inconnu ou pas un lien : retourne un objet
_model (vide)
        //          si le champ est un lien, mais vide, ou pas d'bjet en
face : le bon objet, mais pas chargé
```

```
        // At-on déjà la cible (dans $this->targets)
        if (isset($this->targets[$fieldName])) {
            return $this->targets[$fieldName];
        }


        // Est-ce que c'est un lien ?
        if ( ! isset($this->links[$fieldName])) {
            // Ce n'est pas un lien : on retourne un objet de la classe
_model
            $this->targets[$fieldName] = new _model();
            return $this->targets[$fieldName];
        }

        // c'est un lien : l'objet pointé est de la classe indéiquée dans
$this->links[$fieldName]
        $nomClasse = $this->links[$fieldName];
        $this->targets[$fieldName] = new
$nomClasse($this->get($fieldName));

        return $this->targets[$fieldName];

    }

    function id() {
        // Rôle : récupérer l'id
        // paramètres : néant
        // Retour : l'id ou 0 (un nombre entier)

        //L'id est stocké dans l'atttrbut id
        return $this->id;

    }



    // Setters : donne des valeurs aux paramètres
    // Au lieu de $contact->setNom("Durand), on va avoir une syntaxe
$contact->set("nom", "Durand")
    function set($fieldName, $value) {
```

```
        // Rôle : changer / initialiser la valeur d'un attribut
        // Paramètres : quel attrribut, quelle nouvelle valeur
        //      $fieldName : nom de l'attribut
        //      $value : nouvelle valeur
        // Retour : true si accepté, false sinon

        // Il faut stocker la valeur à l'index correspondant à l'attribut de nom $fieldName, dans $this->values
        $this->values[$fieldName] = $value;

        // On retourne true (on a pas d'infos pour vérifier si la valeur est valide, on ne peut que l'accepter)
        return true;
    }




    // Méthode de synchronisation avec la BDD

    function load($id) {
        // Rôle : chargement de l'objet (de ses attributs) depuis une ligne de la base de données
        // Paramètres :
        //      $id : id du contact à charger
        // Retour : true si on l'a trouvé, false sinon

        // Passe la requête : SELECT champs FROM table WHERE id = monId
        // Construire la requête SELECT `champ1`, `champ2`, .... FROM `table` WHERE id = :id
        // On valorise :id dans un tableau pour l'excéution
        $sql = "SELECT " ;
        // Ajouter tous les noms de champs, encadrés par `, séparés par ,
        // Le noms des champs : ils sont dans l'attribut fields (tableau $this->fields)
        // On met les noms de tous les champs, encadrés par ` `, séparés par une virgule

        // faire un tableau avec les morceux de texte à séparer
```

```php
        // Utiliser la fonction implode pour générer le texte compoésé de tous les éléments séparés un texte donné

        // On génère un tableau composés des noms des champs encadrés par ` `
        $tableau = [];
        foreach($this->fields as $nomChamp) {
            $tableau[] = "`$nomChamp`";
        }
        $sql .= implode(", ", $tableau);

        // Ajouter FROM puis  le nom de la table (il est dans $this->table) encadré par `
        $sql .= " FROM `$this->table` ";

        // Ajouter le texte WHERE `id` = :id
        $sql .= " WHERE `id` = :id";

        // Faire le tableau pour valoriser :id
        $param = [ ":id" => $id];

        //      Préparer / exécuter
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);
        if ( ! $req->execute($param)) {
            // On a une erreur de requête (on peut afficher des messages en phase de debug)
            return false;
        }

        // On s'assure que l'on a trouvé une ligne
        $listeExtraite = $req->fetchAll(PDO::FETCH_ASSOC);
        // Si le tabeu $liste est vide (0 elt), c'est qu'on a pas l'id cherché
        if (empty($listeExtraite)) {
            return false;
        }

        // Transfere son résultat dans les valeurs des attributs internes
        // On récupère le premier (et seul) élément
```

```php
        $tab = $listeExtraite[0];

        // Pour chaque champ de l'objet, on valorise $this->values[champ];
        foreach($this->fields as $nomChamp) {
            $this->values[$nomChamp] = $tab[$nomChamp];
        }

        // On renseigne l'id :
        $this->id = $id;

        return true;

    }


    function insert() {
        // Rôle : création du contact courant dans la base de données
        // paramètres : néant (on utilise les attributs de l'objet)
        // retour : true si réussi, false si échoué

        // Créer la requête : INSERT INTO `nomDeLaTable` SET `nomChamp1` =
:nomChamp1, `nomChamp2` = :nomChamp2, ...
        // En paralèle, faire un tableau de valorisation des :nomChampX :
[ ":nomChamp1" => valeurChamp1, ":nomChamp1" => valeurChamp2, ...]

        $sql = "INSERT INTO `$this->table`SET " . $this->makeRequestSet();
        $param  = $this->makeRequestParamForSet();

        // On prépare la requête
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        //  - on exécute cette requête
        if ( ! $req->execute($param)) {
            // Erreur sur la requête
            return false;
        }

        // ne pas oublier d'enregistrer l'id qui a été généré par la BDD
        // il est dnné par la méhode lastInsertId de l'objet $bdd
```

```
        $this->id = $bdd->lastInsertId();

        return true;

    }


    function update() {
        // Rôle : mettre à jour l'objet courant dans la base de données
        // Paramètres : néant
        // Retour : true si réussi, false si échoué

        // On va devoir exécuter une requête SQL de mise à jour (UPDATE)
        //  - Construire le texte de la requête SQL
        //          UPDATE `nomDeLaTable`
        //              SET `nomChamp1` = :nomChamp1, `nomChamp2` =
:noùmChamp2, ...
        //              WHERE `id` = :id
        //      et valoriser les paramètres
        //          [ ":id" => idLigneAModifier, ":nomChamp1" =>
valeurChamp1, ":nomChamp1" => valeurChamp2, ...]
        // On va donc préparer deux variables :
        //      un texte pour la requête ($sql)
        //      un tableau pour les paramètres :xx de cette requête
        //      on construit la chaine : texte UPDATE `nomDeLaTable` SET
(nomDeLaTable est dans l'atribut table de l'objet)
        //      on va ajouter tous les champs : `nomChamp1` = :nomChamp1
(en les séparant par une virgule)
        //      on va mettre la partie finale ( WHERE ìd`= :id)
        // Et pour le tableau : on met l'élément :id (la valeur est dans
l'attrribut id)
        //      et tous les champs (un par un "pour chaque" ) : attribut
:nomChamp, valeur : elle est dans la tableau values
        //                      on peut aussi récupérer la valeur avec la
méthode get(nomChamp)


        $sql = "UPDATE  `$this->table` SET " . $this->makeRequestSet() . "
WHERE `id` = :id ";
        $param = $this->makeRequestParamForSet();
```

```php
        $param[":id"] = $this->id;


        // On prépare la requête
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        //  - on exécute cette requête
        if ( ! $req->execute($param)) {
            // Erreur sur la requête
            return false;
        }

        return true;

    }

    function makeRequestSet() {
        // Rôle : construire la partie d'une requête de mise à jour ou de création valorisant les champs
        // paramètres : néant
        // Retour : le texte à mettre derrère SET dans une requête SQL : `nomChamp1` = :nomChamp1, `nomChamp2` = :noùmChamp2, ...

        // Je n'ai comme information disponible que :
            // - les attributs de la classe
            // - les paramètres de ma méthode (aucun dans ce cas)
            // - les élments que d'autres méthodes peuvent me donner

        // On a des bouts de texte ( `nomChamp` = :nomChamp) à fabriquer (un pour chaque champ ), et à les séparer par ,
        // Une solution est de :
        //          - fabriquer un tableau simple contenant les bouts de texte
        //          - utiliser implode pour générer la chaine de caractère finale avec les séprateurs

        // Fabrique un tableau simplde des bouts de texte ( `nomChamp` = :nomChamp)
        $tableau = $this->makeTableauSimpleSet();
```

```
        // Générer le texte final :
        return implode(", ", $tableau);


    }

    function makeTableauSimpleSet() {
        // Rôle : faire un tableau contenant pour chaque champ, un élément
avec le texte `nomChamp` = :nomChamp
        // paramètres : néant
        // Retour : le tableau décrit ci-dessus

        // Faire un tableau : on part d'un tableau vide
        $result = [];

        // Pour chaque champ : ajouter dans $result un élément `nomChamp`
= :nomChamp
        foreach($this->fields as $nomChamp) {
            // On a le nom du champ dans $nomchamp
            $result[] = "`$nomChamp` = :$nomChamp";
        }
        return $result;


    }

    function makeRequestParamForSet() {
        // Rôle : préparer (et retourner) le tableau de valorisation des
paramètres pour mise à jour des champs
        // Paramètres : néant
        // Retour : le tableau contenant les valeurs associées aux
:nomChamp (pour chaque champ)
        //               [ ":nomChamp1" => valeur1, ":nomChamp2" =>
valeur2, ... ]



        // Je n'ai comme information disponible que :
```

```
            // - les attributs de la classe
            // - les paramètres de ma méthode (aucun dans ce cas)
            // - les élments que d'autres méthodes peuvent me donner

        // On doit faire un tableau, qui a un élément pour chaque champ
(du modèle conceptuel) de la table
        //      pour le chmpa dont le nom est nomChamp, l'élément du
tabelau résultat
        //          à pour valeur la valeur courant du champ
        //          pour index le caractère : suivi du nom du champ
        // Pour chaque champ : il faut parcourir la liste des champs :
attribut fields ($this->fields)
        $result = [];       // On part d'un tableau vide
        foreach($this->fields as $nomChamp) {
            // On doit ajouter dans le tableau result l'index :nomChamp
avecla valeur du champ
            // ON doit construire $result[":nomChamp"] = valeurDuChamp;
            $index = ":$nomChamp";
            // Valeur : elle est dans le tableau des valeurs, l'attribut
values ($this->values)
            // Si on a une valeur pour $nomChamp, on crée l'élément de
tableau avec cette valeur,
            // Sinon, on crée avec null
            if (isset($this->values[$nomChamp])) {
                $result[$index] = $this->values[$nomChamp];
            } else {
                $result[$index] = null;
            }
        }

        return $result;

    }

    function delete() {
        // Rôle : supprimer l'objet courant dans la base de données
        // Paramètres : néant
        // Retour : Ceux qui sont sur les cp

        // On va devoir exécuter la requête de suppression d'une ligne
```

```
        // En PHP, pour faire une requête sur la BDD :
        //  - on construit le texte de la requête en langage en langae SQL
        // La requête à construire : DELETE FROM `nomDeLaTable` WHERE `id`
= lidQueJeVeux
        // Les parties "variables" sont :
        //      - nomDeLaTable : elle est das l'attribut table de l'objet
courant
        //      - lidQueJeVeux : il est dans mon attribut id
        // La syntaxe pour récupérer un attribut de l'objet courant :
$this->nomAttribut
        $sql = "DELETE FROM `$this->table` WHERE `id` = :id";
        $param = [":id" => $this->id];

        //  - on préparer un objet requête
        // Cela s'aplique à l'objet base de données : il est en variable
static, c'est $bdd
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        //  - on exécute cette requête
        if ( ! $req->execute($param)) {
            // Erreur sur la requête
            return false;
        }
        //  - on exploite le résultat
        // Marquer le fait que cette objet n'est plus dans la BDD : la
régle choisie est de mettre l'id à 0
        $this->id = 0;

        return true;

    }

    function listAll(...$tris) {
        // Rôle : donner la liste de tous les objets de cette calsse
(depuis la BDD)
        // paramètres : gérer les critères de tri
        //          "+/-nomChamp", "+/-nnomChamp", ....
        //      autant de paramètres que de critères de tri,
```

```
        //     chaque paramètre est le nom du champ précédé de - pour un
tri descedant,
        //              optionnellement de + pour un tri ascendant
        //      $obj->tri("+nom", "+prenom")
        //  $tris : on donne mles paramètres séparés par une virgle à
l'appel,
        //        on récupère un tableau simple dans la fonction
        // retour : liste d'objet de la classe courante, indexées par les
id

        // Construire la requête SQL, et ses paramètres
        // SELECT `nomChamp1`, `nomChamp2`, ... FROM `nomTable`
        //      ORDER BY tri1 ASC/DESC, ....

        $sql = "SELECT ";
        // Construire la liste des champs encadrés par `
        $tableau = ["`id`"];
        foreach($this->fields as $nomChamp) {
            $tableau[] = "`$nomChamp`";
        }
        $sql .= implode(", ", $tableau) . " FROM `$this->table`";

        // Construire la liste des critères de tri
        $tabOrder = [];
        foreach($tris as $tri) {
            // tri : +nomChamp ou - nomChamp ou nomChamp
            $car1 = substr($tri, 0, 1);
            if ($car1 === "-") {
                $ordre = "DESC";
                $nomField = substr($tri, 1);
            } else if ($car1 === "+") {
                $ordre = "ASC";
                $nomField = substr($tri, 1);
            } else {
                $ordre = "ASC";
                $nomField = $tri;
            }
            $tabOrder[] = "`$nomField` $ordre";
        }
```

```php
        if (!empty($tabOrder))  $sql .= " ORDER BY " . implode(",",
$tabOrder);

        // préparer / exécuter
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);
        if ( ! $req->execute()) {
            // Echec de la requête
            return [];
        }
        // Construire le tableau résultat
        $result = [];
        // tant que j'ai une ligne de résultat de la requête à lire
        while ($tabObject = $req->fetch(PDO::FETCH_ASSOC)) {
            // "transférer" $tabObject en objet de la classe courante
            // Récupération du nom de la classe de l'objet courant
            $classe = get_class($this);
            $obj = new $classe();
            // Charger l'objet
            $obj->loadFromtab($tabObject);
            // ON ajoute cela dans $result
            $result[$obj->id()] = $obj;
        }
        return $result;
    }

    function loadFromTab($tab) {
        // Rôle : initialiser l'objet (complètement) à partir d'un tableau
de données (simialire à celui récupéré par fetch)
        // Paramètres :
        //      $tab : tableau valorisant les champs du MPD
        // Retour : true si ok, false sinon

        if (isset($tab["id"])) $this->id = $tab["id"];
        foreach($this->fields as $nomChamp) {
            if (isset($tab[$nomChamp]))
                $this->values[$nomChamp] = $tab[$nomChamp];
        }
    }
```

```
}
```

Session.php

```php
<?php

/*



Fonctions de gestion de la session

On a une superglobale $_SESSION (on en fait un tableau)
   - quand PHP est fini : elle est enregistrée
   - quand on lance session_start (un controleur) : elle est restaurée

   On va gérer cette varaible de la manière suivante :
    - l'index id contiendra 0 (ou n'existera pas) quand aucun utilisateur
n'est connecté
    - si un utilisateur est connecté $_SESSION["id"] contient l'id de
l'utilisateur

    quand un utilisateur est connecté,
        on va stocker l'objet associé dans la variable globale
$utilisateurConnecte
*/

function session_activation() {
    // Rôle : acrtive le mécanisme de session
    // paramètres : néant
    // retour : true si on est connecté, false sinon

    // démarrer le maécanisme
    session_start();



    // Si un utilisateur est connecté :
    if (session_isconnected()) {
        //   - charger l'objet utilisateur connecté
        global $personnageConnecte;
```

```
        $personnageConnecte = new personnage(session_idconnected());
        //   - vérifier qu'il est actif, encore autorisé, etc....
        // ....
    }



    // Retourner si on est connecté ou pas
    return session_isconnected();

}

function session_isconnected() {
    // Rôle : dire si il y a une connexion active ou pas
    // Paramètres : néant
    // Retour : tue is on est connect, false sinon

    return ! empty($_SESSION["id"]);
}

 function session_idconnected() {
    // Rôle : donné l'id de l'utilisateur connexté
    // Paramètres : néant
    // Retour : l'id ou 0

    if (session_isconnected()) {
        return $_SESSION["id"];
    } else {
        return 0;
    }

 }

 function session_userconnected() {
    // Rôle : donné l'objet correspondant à l'utilisateur connecté
    // Paramètres : néant
    // Retour : un objet de la calsse qui gère les utilisateurs de l'appli

    if (session_isconnected()) {
        global $personnageConnecte;
```

```
        return $personnageConnecte;
    } else {
        return new personnage();
    }


}

function session_deconnect() {
    // Rôle : déconnecter la session courante
    // paramètres : néant
    // Retour : true

    $_SESSION["id"] = 0;
}

function session_connect($id) {
    // Rôle : connecter un utilisateur
    // paramètres :
    //      $id : id de l'utilisateur connecté
    // Retour : true

    $_SESSION["id"] = $id;

}
```

Verif_connexion.php

```
<?php
/*

Code à inclure dans les controleurs qui ont besoin de la connexion

*/


// Si on n'est pas connexté : rediriger / afficher le formulaire de
connexion
if ( ! session_isconnected()) {
    include "templates/pages/page_connexion.php";
    exit;
```

```
}
```

Modeles

Emojis.php

```php
<?php

// création du tableau des emogis

$emogis = [1 => "👻", 2 => "😇", 3 => "😈", 4 => "👾", 5 => "👽", 6 => "👹", 7 => "👺", 8 => "💩", 9 => "💀", 10 => "☃"];
```

Historique.php

```php
<?php

class historique extends _model {

    protected $table = "historique";
    protected $fields = ["attaquant", "defenseur", "action", "result", "riposte", "detail", "agi_att", "pv_att", "agi_def", "pv_def", "date"];
    protected $links = ["attaquant" => "personnage", "defenseur" => "personnage"];

    function histoPerso($idUser, $date){
        // Role : récupère l'historique des actions en lien avec l'utilisateur depuis sa dernière connexion (sauf si création de compte)
        // parametre : îdUser - id de l'utilisateur  connecté
        // retour : $histo - tableau d'objet historique indéxé par l'id

        // construction
        $sql = "SELECT `id`, `attaquant`, `defenseur`, `action`, `result`, `riposte`, `detail`, `agi_att`, `pv_att`, `agi_def`, `pv_def`, `date` FROM `historique` WHERE (`attaquant` = :id OR `defenseur` = :id) AND `date` >= :dat ORDER BY `date` DESC";
        $param = [":id" => $idUser, ":dat" => $date];

        // préparation
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        //execution
```

```
        if(!$req->execute($param)){
            // erreur de syntaxe - code de debug
            echo "echec sql : $sql";
            print_r($param);
            return [];
        }

        // récupération
        $histos = [];
        while($result = $req->fetch(PDO::FETCH_ASSOC)){
            $histo = new historique();
            $histo->loadFromTab($result);
            $histos[$histo->id()] = $histo;
        }


        // retour
        return $histos;
    }

    function lastAttack($idUser){
        // role : récupère la date de la dernière attaque
        // parametre : id de l'utilisateur
        // retour : $date - date de la derniere attaque

        // construction
        $sql = "SELECT `date` FROM `historique` WHERE `attaquant` = :id
ORDER BY `date` DESC LIMIT 1";
        $param = [":id" => $idUser];

        // préparation
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        // execution
        if(!$req->execute($param)){
            // erreur syntaxe : code de debug
            echo "Echec SQL : $sql";
            print_r($param);
            return "";
```

```
        }

        // récupération
        $result = $req->fetchAll(PDO::FETCH_ASSOC);

        // retour
        return $result[0];
    }
}
```

Personnage.php

```
<?php

class personnage extends _model {

    protected $table = "personnage";
    protected $fields = ["pseudo", "emogi", "mdp", "pv", "for", "agi",
"res", "pos", "date"];

    function verifPseudo($pseudo){
        // role : vérifie si le pseudo testé existe deja dans la base
        // parametre : $pseudo - pseudo à vérifier
        // retour : true si pseudo dispo / false sinon

        // construction
        $sql = "SELECT `pseudo` FROM `personnage` WHERE `pseudo` =
:pseudo";
        $param = [":pseudo" => $pseudo];
        // préparation
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        // execution
        if(!$req->execute($param)){
            // erreur syntaxe, code de debug
            echo "echec sql : $sql";
            print_r($param);
            return false;
        }
```

```
        //récupération
        if($req->fetch(PDO::FETCH_ASSOC)){
            // si il y a une réponse c'est qu'un pseudo existe
            return false;
        }else{
            // sinon pseudo libre
            return true;
        }

    }

    function defend($att){
        // role compare la force d'un joueur attaquand avec les attribut
d'un joueur defendand
        // parametres : $for - force de l'attaquant
        // retour : esquive / riposte / victoire / defaite

        // si valeur nom numérique, on convertit
        if(!is_numeric($this->get("agi"))){
            $agi = intval($this->get("agi"));
        }else{
            $agi = $this->get("agi");
        }
        if(!is_numeric($this->get("for"))){
            $for = intval($this->get("for"));
        }else{
            $for = $this->get("for");
        }
        if(!is_numeric($this->get("res"))){
            $res = intval($this->get("res"));
        }else{
            $res = $this->get("res");
        }

        if($agi - $att > 2){
            // Si notre agilité dépasse la force d'attaque d'au moins 3
points, on esquive.
            return "esquive";
        }else if($for > $att){
```

```
            // Si notre force est supérieure strictement à celle de
l'attaque, on riposte
            return "riposte";
        }else if($res >= $att){
            // si notre résistance est supérieure ou égale à la force de
l'attaque, on gagne
            return "defaite";
        }else{
            // si elle est inférieure, on le perd
            return "victoire";
        }
    }

    function setMdp($mdp){
        // role : crypte mdp avant de le charger dans la bdd
        // parametres : $mdp - le mdp à crypter
        // retour : true/false

        // cryptage
        $hash = password_hash($mdp, PASSWORD_DEFAULT);

        // stockage
        $this->values["mdp"] = $hash;

        // retour
        return true;
    }

    function connexion($pseudo, $mdp){
        // role : recherche dans la base de donnée un utilisateur
correspondant aux critere de recherche
        // parametre : $log = mail de l'utilisateur
                    // $pwd = pwd de l'utilisateur
        // retour : l'id de l'utilisateur

        // construction
        $sql = "SELECT `id`, `mdp` FROM `personnage` WHERE `pseudo` =
:pseudo";
        $param = [":pseudo" => $pseudo];
```

```
        // préparation
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);
        // éxecution
        if(! $req->execute($param)){
            // erreur syntaxe - code de debug
            echo "Echec sql : $sql";
            print_r($param);
            exit;
        }

        // récupération
        $result = $req->fetch(PDO::FETCH_ASSOC);
        if(isset($result)){
            if(password_verify($mdp, $result["mdp"])){
                $id = $result["id"];
            }else{
                $id = 0;
            }
        }else{
            $id = 0 ;
        }
        $this->id = $id;

        // retour
        return $this->id;
    }

    function listeAdversaire(){
        // Role : récupère la liste des personnage dans la meme sale que
l'utilisateur
        // parametre : this id - id de l'utilisateur connecté
        //             this pos - position de l'utilisateur connecté
        // retour : $histo - tableau d'objet historique indéxé par l'id

        // construction
        $sql = "SELECT `id`, `pseudo`, `emogi`, `pv` FROM `personnage`
WHERE `id` != :id AND `pos` = :pos";
        $param = [":id" => $this->id(), ":pos" => $this->get("pos")];
```

```
        // préparation
        $bdd = static::bdd();
        $req = $bdd->prepare($sql);

        //execution
        if(!$req->execute($param)){
            // erreur de syntaxe - code de debug
            echo "echec sql : $sql";
            print_r($param);
            return [];
        }

        // récupération
        $persos = [];
        while($result = $req->fetch(PDO::FETCH_ASSOC)){
            $personnage = [];
            $personnage["id"] = $result["id"];
            $personnage["pseudo"] = $result["pseudo"];
            $personnage["emogi"] = $result["emogi"];
            $personnage["pv"] = $result["pv"];
            $persos[] = $personnage;
        }

        // retour
        return $persos;
    }
}
```

Tableaux.php

```
<?php

// création du tableau des emogis

$emogis = [1 => "👻", 2 => "👼", 3 => "😈", 4 => "👾", 5 => "👽", 6 =>
"👹", 7 => "👺", 8 => "💩", 9 => "💀", 10 => "☃"];

$noms = [
     0 => "à l'entrée",
     1 => "à la forêt enchantée",
```

```
    2 => "à la caverne de glace",
    3 => "au temple de feu",
    4 => "à la bibliothèque abandonnée",
    5 => "au jardin souterrain",
    6 => "à la chambre des trésors",
    7 => "aux marais toxique",
    8 => "au cimetière hantée",
    9 => "à la salle du boss",
    10 => "à la sortie",
    11 => "à la salle des âmes perdues"
    ];

$fonds = [
    0 => "entree",
    1 => "un",
    2 => "deux",
    3 => "trois",
    4 => "quatre",
    5 => "cinq",
    6 => "six",
    7 => "sept",
    8 => "huit",
    9 => "neuf",
    10 => "sortie",
    11 => "mort"
];

$pieces = [
    0 => "🏰 Entrée 🏰",
    1 => "🌳 Forêt enchantée 🌳",
    2 => "❄️ Caverne de glace ❄️",
    3 => "🔥 Temple du feu 🔥",
    4 => "📚 Bibliothèque abandonnée 📚",
    5 => "🌷 Jardin souterrain 🌷",
    6 => "💰 Chambre des trésors 💰",
    7 => "☠️ Marais toxique ☠️",
    8 => "👻 Cimetière hanté 👻",
    9 => "👹 Salle du boss 👹",
    10 => "🗝️ Sortie 🗝️",
    11 => "⚜️ Salle des âmes perdues ⚜️"
```

```
    ];
```

Templates
Fragments
Adversaires.php

```php
<?php

// fragment liste des adversaires présent dans la pièce

if($perso->get("pos") > 0 && $perso->get("pos") < 10){
    $adversaires = $perso->listeAdversaire();
    if(isset($adversaires)){
        foreach($adversaires as $id => $adv){
        ?>
        <button data-idadv="<?= $adv["id"] ?>" class='w120p fs10 btnActif'><?= $emogis[$adv["emogi"]] ?> <?= $adv["pseudo"] ?> - <?= $adv["pv"] ?></button>
        <?php
        }
    }else{
        echo "<p class='mt16 fs16 yellow txt-center'>Tu n'as pas d'adversaires dans cette pièce</p>";
    }
}else if($perso->get("pos") == 0){
    echo "<p class='fs12 yellow txt-center'>Bienvenu dans 'Epic Clash : Lost Dungeon', entre dans le donjon, combat tes adversaires et soit le premier à atteindre la sortie.</p>";
    echo "<div class='flex j-center w100'><button class='fs10 wfit btnActif'>Règles du jeu</button></div>";
}else if($perso->get("pos") == 10){
    echo "<p class='fs12 yellow txt-center'>Bravo ! Tu as atteint la sortie du donjon, ton aventure se termine ici, crée un nouveau personnage pour rejouer</p>";
    echo "<div class='flex j-center w100'><a href='afficher_form_creation.php'><button class='fs10 wfit btnActif'>Créer un nouveau personnage</button></a></div>";
}else if($perso->get("pos") == 11){
    echo "<p class='fs12 yellow txt-center'>Ton personnage a succombé à une attaqe. C'est la fin de ton aventure, crée un nouveau personnage pour rejouer</p>";
```

```
    echo "<div class='flex j-center w100'><a
href='afficher_form_creation.php'><button class='fs10 wfit btnActif'>Créer
un nouveau personnage</button></a></div>";
}
```

Header.php

```
<div class="containFond">
        <div class="fond <?= $fonds[$pos] ?>"></div>
    </div>
<div class="titre flex a-center">
    <h1 class="txt-center yellow w100 fs20">💥 Epic Clash : Lost Dungeon
💥</h1>
</div>
```

Histo.php

```
<?php

// fragment liste l'historique du joueur
// parametres : $perso - objet utilisateur connecté
//              $histo - liste des historique de l'utilisateur

// pour chaque objet historique de la liste
foreach($histos as $id => $histo){
    //je récupère la date de l'histo
    $date = $histo->get("date");
    echo "<li class='fs12 yellow'><span class='fs10'>$date</span> :</li>";
    echo "<li class='w100 txt-end fs12 yellow'>";
    // si je suis l'attaquant
    if($histo->get("attaquant") === $idUser){
        // si action = creation
        if($histo->get("action") === "creation"){
            $emogi = $perso->get("emogi");
            $logo = $emogis[$emogi];
            $pseudo = $perso->get('pseudo');
            echo "création de ton personnage ($logo $pseudo)";
        }else if($histo->get("action") === "mouvement"){
            // si action = mouvement
            $piece = $histo->get("detail");
            $nomPiece = $noms[$piece];
            // si result = suivante
```

```
            if($histo->get("result") === "suivante"){
                // si piece = 10
                if($piece == "10"){
                    echo "<span class='green'>tu accède à la sortie du
donjon, le jeu est terminé !</span>";
                }else{
                //sinon
                echo "⇒ tu accède $nomPiece (<span class='red'>-$piece
⚡</span>)";
                }
            }else if($histo->get("result") === "precedente"){
                // si result = precedente
                echo "⇐ tu accède $nomPiece (<span class='green'>+$piece
🩸</span>)";
            }
        }else if($histo->get("action") === "attribut"){
            // si action = attributs
            // si result = forToRes
            if($histo->get("result") === "forToRes"){
                echo "modification de tes attributs (⚔️ ⇒ 🛡️, <span
class='red'>-3 ⚡</span>)";
            }else if($histo->get("result") === "resToFor"){
            // si result = resToFor
                echo "modification de tes attributs (🛡️ ⇒ ⚔️, <span
class='red'>-3 ⚡</span>)";
            }
        }else if($histo->get("action") === "attaque"){
            // si action = attaque (j'attaque)
            // je récupère les info de l'adversaire, j'affiche que je
l'attaque
            $adv = $histo->getTarget("defenseur");
            $advPseudo = $adv->get("pseudo");
            $advEmogi = $emogis[$adv->get("emogi")];
            echo "tu attaque $advEmogi $advPseudo, ";
            // si result = esquive
            if($histo->get("result") === "esquive"){
                echo "il esquive, <span class='orange'>match nul </span>";
                if($histo->get("detail") === "forTores"){
                    echo "(⚔️ ⇒ 🛡️)";
                }
```

```
            }else if($histo->get("result") === "riposte"){
            // si result = riposte
                echo "il riposte, ";
                if($histo->get("riposte") === "esquive"){
                    echo "tu esquives, <span class='orange'>match nul
</span> (<span class='red'>-1 ⚡</span>)";
                }else if($histo->get("riposte") === "victoire"){
                // si result = victoire
                    echo "tu <span class='green'>gagne</span> (";
                    $agi = $histo->get("agi_att");
                    if($agi != 0){
                        echo "(<span class'green'>+$agi</span> ⚡)";
                    }
                    $pv = $histo->get("pv_att");
                    if($pv != 0){
                        echo " (<span class'green'>+$pv</span> 🩸)";
                    }
                    echo ")";
                    if($histo->get("pv_def") == 0){
                        echo " il est mort";
                    }
                }else if($histo->get("riposte") === "defaite"){
                // si result = defaite
                    echo "tu <span class='red'>perd</span> (<span
class='red'>-1</span> 🩸)";
                }
            }else if($histo->get("result") === "victoire"){
            // si result = victoire
                echo "tu <span class='green'>gagne</span> le combat (";
                $agi = $histo->get("agi_att");
                if($agi != 0){
                    echo "<span class'green'>+$agi</span> ⚡";
                }
                $pv = $histo->get("pv_att");
                if($pv != 0){
                    echo " <span class'green'>+$pv</span> 🩸";
                }
                echo ")";
                if($histo->get("pv_def") == 0){
                    echo " il est mort";
```

```
                }
            }else if($histo->get("result") === "defaite"){
            // si result = defaite
                $pv = $histo->get("pv_att");
                echo "tu <span class='red'>perds</span> le combat (<span
class='red'>-1</span> 🩸)";
            }
        }
    }else if($histo->get("defenseur") === $idUser){
        // si je suis defenseur
        // je récupère les info de l'adversaire, j'affiche que je
l'attaque
        $adv = $histo->getTarget("attaquant");
        $advPseudo = $adv->get("pseudo");
        $advEmogi = $emogis[$adv->get("emogi")];
        // si action = attaque (je me fais attaquer)
        if($histo->get("action") === "attaque"){
            echo "$advEmogi $advPseudo t'attaque, ";
            // si result = esquive
            if($histo->get("result") === "esquive"){
                echo "tu esquive, <span class='orange'>match nul</span>
(<span class='red'>-1</span> ⚡)";
            }else if($histo->get("result") === "riposte"){
            // si result = riposte
                echo "tu riposte";
            }else if($histo->get("result") === "victoire"){
            // si result = victoire
                $pv = $histo->get("pv_def");
                echo "tu <span class='red'>perd</span> le combat (<span
class='red'>-$pv</span> 🩸)";
                if($histo->get("pv_def") == 0){
                    echo " tu est mort";
                }
            }else if($histo->get("result") === "defaite"){
            // si result = defaite
                echo "tu <span class='green'>gagne</span> le combat";
            }
        }else if($histo->get("action") === "riposte"){
            // si action = riposte (j'attaque à mon tour)
            // si result = esquive
```

```
            if($histo->get("result") === "esquive"){
                echo "$advEmogi $advPseudo esquive ta riposte, <span
class='orange'>match nul</span>";
                if($histo->get("detail") === "forTores"){
                    echo " (⚔️ ⇒ 🛡️)";
                }
            }else if($histo->get("result") === "victoire"){
                // si result = victoire
                echo "tu <span class='red'>perd</span> la riposte (<span
class='red'>-2</span> 🩸)";
                if($histo->get("pv_def") == 0){
                    echo " tu est mort";
                }
            }else if($histo->get("result") === "defaite"){
                // si result = defaite
                echo "tu <span class='green'>gagne</span> la riposte
(<span class='green'>+1</span> 🩸)";
            }
        }
    }
    echo "</li>";
}
```

Select_emogi.php

```
<?php

// fragment : construit le select du personnage

include "modeles/tableaux.php";


?>

<div class="flex mt16 a-center">
    <label for="emogi" class="yellow mr20 w150p fs12"> Emogi : </label>
    <select class="w130p" name='emogi' id="emogi">
        <?php
        if($emogi_select === ""){
```

```
            ?>
            <option value=''>---</option>
            <?php
        }else{
            ?>
            <option value='<?= $emogi_select ?>'><?=
$emogis[$emogi_select] ?></option>
            <?php
        }
        foreach ($emogis as $id => $emogi) {
            ?>
            <option value="<?= $id ?>"><?= $emogi ?></option>
            <?php
        }
        ?>
    </select>
</div>
```

Pages

Form_creation.php

```
<?php

// template : affiche le formulaire de création d'un personnage
// parametres : aucun

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Création d'un personnage</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="fond creation"></div>
    <?php include "templates/fragments/header.php"; ?>
    <main>
```

```
        <div class="container flex w300p mrlauto j-center">
            <div>
                <h2 class="w100 yellow fs16">Crée ton personnage</h1>
                <form class="mt32 flex column" action="creer.php"
method="post">
                    <div class="flex a-center">
                        <label for="pseudo" class="yellow mr20 w150p
fs12">Pseudo : </label>
                        <input class="w130p" type="text" name="pseudo"
id="pseudo_connexion" value="<?= $pseudo ?>">
                    </div>
                    <?php
                        if($verif === 1){
                            echo "<p class='fs10 red'>Ce pseudo est déjà
utilisé !</p>";
                        }else if($error === 1 && $pseudo === ""){
                            echo "<p class='fs10 red'>Tu dois choisir un
pseudo !</p>";
                        }
                        include "templates/fragments/select_emogi.php";
                        if($error === 1 && $emogi_select === ""){
                            echo "<p class='fs10 red'>Tu dois choisir un
emogi !</p>";
                        }
                    ?>
                    <div class="flex mt16 a-center">
                        <label for="mdp" class="yellow mr20 w150p
fs12">Password : </label>
                        <input class="w80p" type="password"
name="mdp_creation" id="mdp_creation" value="<?= $mdp ?>">
                        <p id="btnMdpCrea" class="fs12 ml16 w34p
txt-center btnActif">👁️</p>
                    </div>
                    <?php
                        if($error === 1 && $mdp === ""){
                            echo "<p class='fs10 red'>Tu dois choisir un
mot de passe !</p>";
                        }
                    ?>
                    <div class="flex mt16 a-center">
```

```
                        <label class="yellow mr20 w150p fs12">Confirme
password : </label>
                        <input class="w80p" type="password"
name="mdpVerif_creation" id="mdpVerif_creation" value="<?= $mdpVerif ?>">
                        <p id="btnMdpVerif" class="fs12 ml16 w34p
txt-center btnActif">👁️</p>
                    </div>
                    <?php
                        if($error === 1 && $mdpVerif === ""){
                            echo "<p class='fs10 red'>Tu dois confirmer
ton mot de passe !</p>";
                        }else if($error === 1 && $mdpVerif !== $mdp){
                            echo "<p class='fs10 red'>Tu t'es trompé dans
la confirmation de ton mot de passe !</p>";
                        }
                    ?>
                    <div class="carac flex column">
                        <p class="yellow mt32 fs16">Caractéristiques de
ton personnage :</p>
                        <p class="yellow mt8 fs12">Tu dispose de 15 points
à répartir entre les différentes caractéristiques (minimum 3 points et
maximum 10 points par caractéristique)</p>
                        <div class="flex a-center mt16">
                            <label for="res" class="yellow mr60 w150p
fs12">🛡️ Résistance (RES) : </label>
                            <input type="number" name="res" id="res"
min="3" max="10" value="<?= $res ?>">
                        </div>
                        <div class="flex a-center mt16">
                            <label class="yellow mr60 w150p fs12">⚔️ Force
(FOR) : </label>
                            <input type="number" name="for" id="for"
min="3" max="10" value="<?= $for ?>">
                        </div>
                        <div class="flex a-center mt16">
                            <label class="yellow mr60 w150p fs12">⚡
Agilité (AGI) : </label>
                            <input type="number" name="agi" id="agi"
min="3" max="10" value="<?= $agi ?>">
                        </div>
```

```
                        <?php
                            if($carac !== 15){
                                echo "<p class='mt8 fs10 red'>Le total des
caractéristiques doit être égal à 15 ! (ni plus, ni moins)</p>";
                            }
                        ?>
                    </div>
                    <input class="wfit mt32 fs12 btnActif" type="submit"
value="Créer mon personnage">
                </form>
            </div>
            <div class="w100 flex j-end"><a
href="afficher_jeu.php"><button class="wfit mt48 fs12
btnActif">Annuler</button></a></div>
        </div>
    </main>
    <script src="js/form_crea.js" defer></script>
</body>
</html>
```

Jeu.php

```
<?php

// template : affiche l'ecran de jeu
// parametres : $perso : objet personnage avec carac - pseudo - pv - for -
agi - res - pos - date)
//              $histo : tableau d'objet historique indexé par l'id
//              $adversaire : liste des joueur présent dans la pièce"
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jeu</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="containFond">
        <div class="fond <?= $fonds[$pos] ?>"></div>
```

```
    </div>
    <?php include "templates/fragments/header.php"; ?>
    <main class="w300p mrlauto">
        <div class="perso">
            <div class="w100 flex j-between a-center">
                <div class="flex j-between a-center">
                    <a href="deconnecter.php"><button class="fs10 w20p
btnActif">🚪</button></a>
                    <a href="afficher_regle.php"><button class="fs10 w20p
btnActif">📜</button></a>
                </div>
                <h2 class="fs20 yellow"><?= $emogis[$perso->get("emogi")]
?> <?= $perso->get("pseudo") ?></h2>
                <p id="pv" class="fs16">🩸 : <?= $perso->get("pv")
?>/100</p>
            </div>
            <ul class="w100 flex j-between mt16 a-center">
                <li id="for" class="fs16 yellow">⚔️ : <?=
$perso->get("for") ?>/15</li>
                <li id="res" class="fs16 yellow">🛡️ : <?=
$perso->get("res") ?>/15</li>
                <li id="agi" class="fs16 yellow">⚡ : <?=
$perso->get("agi") ?>/15</li>
            </ul>
            <div class="w100 attribut flex j-center gap32 mt16 a-center">
                <button id="forToRes" class="fs10 w75p">⚔️ ⇒ 🛡️</button>
                <p class="yellow fs10">-3 ⚡</p>
                <button id="resToFor" class="fs10 w75p">🛡️ ⇒ ⚔️</button>
            </div>
        </div>
        <div class="piece mt32">
            <div class="flex j-between">
                <p id="ctPv" class="yellow fs10 a-end">+<?= $preced ?>
🩸</p>
                <h2 id="piece" class="fs14 yellow wfit txt-center"><?=
$pieces[$pos] ?></h2>
                <p id="ctAgi" class="yellow fs10 a-end">-<?= $suiv ?>
⚡</p>
            </div>
            <div class="flex j-between mt16">
```

```
                <button id="preced" class="fs8 w140p"><?= $pieces[$preced]
?></button>
                <button id="suiv" class="fs8 w140p"><?= $pieces[$suiv]
?></button>
            </div>
        </div>
        <div class="mt32">
            <h2 class="fs12 yellow w100 txt-center">Tes adversaires dans
la pièces :<br>(click pour attaquer)</h2>
            <div class="containerAdv mt16">
                <div id="adv" class="flex j-between gap8">
                    <?php include "templates/fragments/adversaires.php";
?>
                </div>
            </div>
        </div>
        <div class="histo mt32">
            <h2 class="fs12 yellow w100 txt-center">Historique des actions
:</h2>
            <div class="containerHisto mt16">
                <div class="listContainer">
                    <ul id="histo">
                        <?php include "templates/fragments/histo.php"; ?>
                    </ul>
                </div>
            </div>
        </div>
    </main>
    <script src="js/combat.js" defer></script>
</body>
</html>
```

Page_connexion.php

```
<?php

// template : affiche la page de connexion
// parametre aucun
```

```
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Page de connexion</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="fond accueil"></div>
    <?php include "templates/fragments/header.php"; ?>
    <main>
        <div class="container flex w300p mrlauto">
            <div>
                <h2 class="w100 yellow fs16">Connexion</h1>
                <form class="mt32 flex column" action="connecter.php"
method="post">
                    <div class="flex">
                        <label class="yellow w100p fs12">Pseudo : </label>
                        <input class="w130p" type="text" name="pseudo"
id="pseudo_connexion">
                    </div>
                    <div class="flex mt16 a-center">
                        <label class="yellow w100p fs12">Password :
</label>
                        <input class="w80p" type="password" name="mdp"
id="mdp_connexion">
                        <p id="btnMdpConnect" class="fs12 ml16 w34p
txt-center btnActif">👁️</p>
                    </div>
                    <input class="mt32 wfit fs12 btnActif" type="submit"
value="Me connecter">
                </form>
                <?php
                if($error == 1){
                    ?>
                    <p class="mt8 fs10 red">Il y a une erreur de saisie
sur ton pseudo ou ton mot de passe</p>
```

```
                    <?php
                }
                ?>
            </div>
            <div class="w100 flex j-end"><a class="mt80"
href="afficher_regle.php"><button class="fs10 btnActif">📜 Règles du
jeu</button></a></div>
            <div class="flex j-end w300p mrlauto mt80">
                <p class="w100 txt-end yellow fs12">Pas encore de
personnage ?</p>
                <div class="w100 flex j-end"><a
href="afficher_form_creation.php"><button class="mt16 wfit fs12
btnActif">Créer un personnage</button></a></div>
            </div>
        </div>
    </main>
    <script src="js/connexion.js" defer></script>
</body>
</html>
```

Regles.php

```
<?php

// fragment pour les regles

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Règles du jeu</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="containFond">
        <div class="fond regle"></div>
    </div>
    <?php include "templates/fragments/header.php"; ?>
```

```
    <main class="w300p mrlauto yellow">
        <div class="flex j-between a-center">
            <h2 class="fs20">Règles du jeu</h2>
            <a href="afficher_jeu.php"><button class="fs10
btnActif">↵</button></a>
        </div>
        <h3 class="mt32">TRANSFORMER UN POINT (FORCE OU RÉSISTANCE)</h3>
        <p class="mt8 fs10">La transformation d’un point de force en point
de résistance, ou réciproquement, consomme 3 points d’agilité.</p>
        <p class="mt8 fs10">Cela ne permet pas de dépasser 15 points de
force ou 15 points de résistance.</p>
        <h3 class="mt32 fs14">DÉPLACEMENTS</h3>
        <h4 class="mt16 fs12">Déplacement vers l'avant :</h4>
        <p class="mt8 fs10">Un déplacement vers l'avant (dans la pièce
suivante ou la zone de sortie) consomme des points d'agilité : il faut
autant de
        points d'agilité que le numéro de la pièce à atteindre, et 10
points pour passer dans la zone de sortie.</p>
        <p class="mt8 fs10">Si on n'a pas assez de points d'agilité, on
n'a pas accès à la pièce suivante</p>
        <h4 class="mt16 fs12">Déplacement vers l'arrière :</h4>
        <p class="mt8 fs10">Un déplacement vers l'arrière (dans la pièce
précédente ou la zone d'entrée) est toujours possible et ne consomme pas
de points d'agilité.</p>
        <p class="mt8 fs10">On peut le faire même avec zéro point
d'agilité.</p>
        <p class="mt8 fs10">On gagne alors en points de vie le numéro de
la pièce atteinte.</p>
        <h3 class="mt32 fs14">RESTER DANS UNE PIÈCE SANS RIEN FAIRE</h3>
        <p class="mt8 fs10">Si on reste dans une pièce sans rien faire
(sans attaquer) pendant 3 secondes, on récupère 1 point d'agilité.</p>
        <p class="mt8 fs10">Cela ne s'applique pas à la zone d'entrée.</p>
        <p class="mt8 fs10">Attention : les points d'agilité sont
plafonnés à 15 !</p>
        <h3 class="mt32 fs14">ATTAQUER</h3>
        <p class="mt8 fs10">Lorsqu'on clique sur un personnage (donc
présent dans la même pièce), cela signifie qu'on l'attaque.</p>
        <p class="mt8 fs10">L'attaque est alors automatique, et se fait
avec une force déterminée et que l'on ne peut pas choisir (cette force
```

```
        d’attaque est utilisée pour déterminer le déroulement du combat,
voir chapitre suivant) : la force de l’attaque est la force
        de l’attaquant.</p>
        <p class="mt8 fs10">Si l’adversaire esquive et que l’on a 10
points de force ou plus, un point de force devient un point de
résistance.</p>
        <p class="mt8 fs10">Si on gagne le combat, on récupère un point
d'agilité (ça motive ! ), ou un point de vie si on a déjà 15 points
d'agilité.</p>
        <p class="mt8 fs10">Si on gagne le combat et que en plus l'on tue
l'adversaire, on récupère en plus les points de vie qui lui restaient
juste
        avant le combat.</p>
        <p class="mt8 fs10">Si on perd le combat : on perd 1 point de
vie.</p>
        <h3 class="mt32 fs14">SUBIR UNE ATTAQUE</h3>
        <p class="mt8 fs10">Lorsque l'on subit une attaque d'une certaine
force, on se défend, voire on riposte.</p>
        <p class="mt8 fs10">On peut donc subir une attaque alors même que
l'on n'est pas connecté.</p>
        <p class="mt8 fs10">Cette opération est automatique (et apparait
dans l'historique de l'utilisateur).</p>
        <p class="mt8 fs10">Si notre agilité dépasse la force d'attaque
d'au moins 3 points, on esquive. Personne n'a alors gagné ou perdu le
combat,
        et on perd 1 point d'agilité.</p>
        <p class="mt8 fs10">Si notre force est supérieure strictement à
celle de l'attaque, on riposte : voir ci-après la riposte. On gagne le
combat et
        un point de vie si on gagne la riposte, on perd le combat et 2
points de vie si on perd la riposte.</p>
        <p class="mt8 fs10">Sinon, on se défend : si notre résistance est
supérieure ou égale à la force de l'attaque, on gagne le combat, si elle
est
        inférieure, on le perd et on perd en points de vie la différence
entre notre résistance et la force de l'attaque.</p>
        <h3 class="mt32 fs14">RIPOSTE</h3>
        <p class="mt8 fs10">La riposte fonctionne comme une attaque, mais
est déclenchée automatiquement.</p>
```

```
        <p class="mt8 fs10">On attaque alors notre attaquant avec toute
notre force disponible, et notre attaquant applique les règles "subir une
        attaque".</p>
        <div class="flex j-end mt32 mb32">
            <a href="afficher_jeu.php"><button class="fs10
btnActif">retour</button></a>
        </div>
    </main>
</body>
</html>
```

Controleurs

Afficher_form_creation.php

```
<?php

// controleur : dde l'affichage du formulaire de création d'un personnage
// parametres : $GET si prb de saisi lors de la créa


// initialisation
require_once "utils/init.php";

// récupération

$verif = isset($_GET["verif"]) ? 1 : 0;
$error = isset($_GET["error"]) ? 1 : 0;
$pseudo = isset($_GET["pseudo"]) && $_GET["pseudo"] !== "" ?
$_GET["pseudo"] : "";
$emogi_select = isset($_GET["emogi"]) && $_GET["emogi"] !== "" ?
$_GET["emogi"] : "";
$mdp = isset($_GET["mdp"]) && $_GET["mdp"] !== "" ? $_GET["mdp"] : "";
$mdpVerif = isset($_GET["mdpVerif"]) && $_GET["mdpVerif"] !== "" ?
$_GET["mdpVerif"] : "";
$for = isset($_GET["for"]) ? $_GET["for"] : 5;
$agi = isset($_GET["agi"]) ? $_GET["agi"] : 5;
$res = isset($_GET["res"]) ? $_GET["res"] : 5;
$carac = $for + $agi + $res;
```

```php
// affichage
include "templates/pages/form_creation.php";
```

Afficher_jeu.php

```php
<?php

// Controleur : Si pas d'idUser : dde affichage de la page de connexion,
si id, dde l'affichage de l'écran de jeu
// parametre : $_SESSION[id] - id de l'utilisateur connecté
//             $_get[error] : si erreur d'authentification


// initialisation

require_once "utils/init.php";

// surveille si le parametre error est dans le get
$error = isset($_GET["error"]) ? $_GET["error"] : 0;

include "utils/verif_connexion.php";

// récupération

$idUser = session_isconnected() ? session_idconnected() : 0;

include "modeles/tableaux.php";

// traitement

$perso = new personnage($idUser);

$pos = $perso->get("pos");
if($pos == 0){
    $preced = 0;
}else{
    $preced = $pos -1;
```

```
}
$suiv = $pos +1;



$historique = new historique();
$histos = $historique->histoPerso($idUser, $_SESSION["last-co"]);

// affichage

include "templates/pages/jeu.php";
```

Afficher_regle.php

```
<?php
// controleur : dde l'affichage des regles du jeu
// parametres aucun

// initialisation
require_once "utils/init.php";

// affichage
include "templates/pages/regles.php";
```

Attaquer.php

```
<?php

// controleur ajax : lance les différentes méthodes permettant de générer
un combat
// parametres : $_session(idUser) - id deu joueur attaquant
//              $_get(idAdv) - id du joueur attaqué
// retour : template fight avec détail du combat ($histo avec histo du
combat)

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;
$idAdv = isset($_GET["idAdv"]) ? $_GET["idAdv"] : 0;
```

```php
// traitement
$attaquant = new personnage($idUser);
$defenseur = new personnage($idAdv);
$histo = new historique();
$date = date("Y-m-d H:i:s");
$histo->set("date", $date);
$histo->set("attaquant", $idUser);
$histo->set("defenseur", $idAdv);
$histo->set("action", "attaque");

// transformation des donnée non numérique en donnée numérique
if(!is_numeric($attaquant->get("for"))){
    $forAtt = intval($attaquant->get("for"));
}else{
    $forAtt = $attaquant->get("for");
}
if(!is_numeric($attaquant->get("pv"))){
    $pvAtt = intval($attaquant->get("pv"));
}else{
    $pvAtt = $attaquant->get("pv");
}
if(!is_numeric($attaquant->get("agi"))){
    $agiAtt = intval($attaquant->get("agi"));
}else{
    $agiAtt = $attaquant->get("agi");
}
if(!is_numeric($attaquant->get("res"))){
    $resAtt = intval($attaquant->get("res"));
}else{
    $resAtt = $attaquant->get("res");
}
if(!is_numeric($defenseur->get("for"))){
    $forDef = intval($defenseur->get("for"));
}else{
    $forDef = $defenseur->get("for");
}
if(!is_numeric($defenseur->get("pv"))){
    $pvDef = intval($defenseur->get("pv"));
}else{
    $pvDef = $defenseur->get("pv");
```

```
}
if(!is_numeric($defenseur->get("agi"))){
    $agiDef = intval($defenseur->get("agi"));
}else{
    $agiDef = $defenseur->get("agi");
}
if(!is_numeric($defenseur->get("res"))){
    $resDef = intval($defenseur->get("res"));
}else{
    $resDef = $defenseur->get("res");
}

// l'attaquand attaque avec sa force, le defenseur se defend
$resultat = $defenseur->defend($forAtt);
if($resultat === "esquive"){
    // si resultat = esquive
    $histo->set("result", "esquive");
    // match nul et def perd 1 agi
    $newAgiDef = $agiDef-1;
    if($newAgiDef >= 0){
        $defenseur->set("agi", $newAgiDef);
        $histo->set("agi_def", -1);
    }
    // si foratt > 10
    if($forAtt > 10){
        // un for devient un res pour l'attaquant
        $histo->set("detail", "forToRes");
        $newForAtt = $forAtt-1;
        $newResAtt = $resAtt+1;
        $attaquant->set("for", $newForAtt);
        $attaquant->set("res", $newResAtt);
    }
}else if($resultat === "riposte"){
    // si resultat = riposte
    // historise
    $histo->set("result", "riposte");
    // le defenseur attaque avec sa force, l'attaquand se defend et je
recup le resultat
    $resultatR = $attaquant->defend($forDef);
    // si resultat = esquive
```

```
    if($resultatR === "esquive"){
        $histo->set("riposte", "esquive");
        // match nul et def perd 1 agi
        $newAgiAtt = $agiAtt-1;
        if($newAgiAtt >= 0){
            $attaquant->set("agi", $newAgiAtt);
            $histo->set("agi_att", -1);
        }
        // si fordef > 10
        if($forDef > 10){
            // un for devient un res pour le defendant
            $histo->set("detail", "forToRes");
            $newForDef = $forDef-1;
            $newResDef = $resDef-1;
            $defenseur->set("for", $newForDef);
            $defenseur->set("res", $newResDef);
        }
    }else if($resultatR === "défaite"){
        // si resultat = defaite - defaite de la riposte dc victoire de
l'attaquant
        $histo->set("riposte", "victoire");
        // def perd 2pv
        // si def pas dead
        if($pvDef-2 > 0){
            $newPvDef = $pvDef-2;
            $defenseur->set("pv", $newPvDef);
            $histo->set("pv_def", -2);
            // si agiatt < 15
            if($agiAtt < 15){
                // agiatt += 1
                $newAgiAtt = $agiAtt+1;
                $attaquant->set("agi", $newAgiAtt);
                $histo->set("agi_att", +1);
            }else{
                // si agiatt = 15
                // pvatt +=1
                $newPvAtt = $pvAtt+1;
                $attaquant->set("pv", $newPvAtt);
                $histo->set("pv_att", 1);
            }
```

```
        }else{
            // si def dead :gagne pvdef
            $defenseur->set("pv", 0);
            $defenseur->set("pos", 11);
            $histo->set("pv_def", 0);
            // si agiatt+pvdef <=15
            if($agiAtt+$pvDef <= 15){
                // agiatt += pvdef
                $newAgiAtt = $agiAtt+$pvDef;
                $attaquant->set("agi", $newAgiAtt);
                $histo->set("agi_att", $pvDef);
            }else{
                // si agiatt+pvdef > 15
                // agiatt += 15-agiatt
                $gainAgiAtt = 15-$agiAtt;
                if($agiAtt < 15){
                    $newAgiAtt = $pvAtt+$gainAgiAtt;
                    $attaquant->set("agi", $newAgiAtt);
                    $histo->set("agi_att", $gainAgiAtt);
                }
                if($pvAtt < 100)
                    // pvatt += pvdef-(15-agiatt)
                    $gainPvAtt = $pvDef-$gainAgiAtt;
                    $newPvAtt = $pvAtt+$gainPvAtt;
                    if($newPvAtt <= 100){
                        $attaquant->set("pv", $newPvAtt);
                        $histo->set("agi_att", $gainPvAtt);
                    }
            }
        }
    }else if($resultatR === "victoire"){
        // résultat = victoire de la riposte dc defaite de l'attaquant
        $histo->set("riposte", "defaite");
        // att perd 1pv
        if($pvAtt > 0){
            $histo->set("pv_att", -1);
            $newPvAtt = $pvAtt - 1;
            $attaquant->set("pv", $newPvAtt);
            if($newPvAtt == 0){
                $attaquant->set("pos", 11);
```

```
                $histo->set("pv_att", 0);
            }
        }
        // defenseur gagne 1pv
        if($pvDef < 100){
            $histo->set("pv_def", 1);
            $newPvDef = $pvDef+1;
            $defenseur->set("pv", $newPvDef);
        }
    }
}else if($resultat === "victoire"){
    // si resultat = victoire
    $histo->set("result", "victoire");
    // def perd foratt-resdef
    // si def pas dead
    $ecartForRes = $forAtt-$resDef;
    $newPvDef = $pvDef-$ecartForRes;
    if($newPvDef > 0){
        $defenseur->set("pv", $newPvDef);
        $histo->set("pv_def", $ecartForRes);
        // si agiatt < 15
        if($agiAtt < 15){
            // agiatt += 1
            $newAgiAtt = $agiAtt+1;
            $attaquant->set("agi", $newAgiAtt);
            $histo->set("agi_att", 1);
        }else if($pvAtt < 100){
            // si agiatt = 15
            // pvatt +=1
            $newPvAtt = $pvAtt+1;
            $attaquant->set("pv", $newPvAtt);
            $histo->set("pv_att", 1);
        }
    }else{
        // si def dead :gagne pvdef
        $defenseur->set("pv", 0);
        $defenseur->set("pos", 11);
        $histo->set("pv_def", 0);
        // si agiatt+pvdef <=15
        if($agiAtt+$pvDef <= 15){
```

```
            // agiatt += pvdef
            $newAgiAtt = $agiAtt+$pvDef;
            $attaquant->set("agi", $newAgiAtt);
            $histo->set("agi_att", $pvDef);
        }else{
            // si agiatt+pvdef > 15
            // agiatt += 15-agiatt
            $gainAgiAtt = 15-$agiAtt;
            $newAgiAtt = $pvAtt + $gainAgiAtt;
            $attaquant->set("agi", $newAgiAtt);
            $histo->set("agi_att", $gainAgiAtt);
            // pvatt += pvdef-(15-agiatt)
            $gainPvAtt = $pvDef-$gainAgiAtt;
            if($pvAtt+$gainPvAtt <= 100){
                $newPvAtt = $pvAtt + $gainPvAtt;
                $attaquant->set("pv", $newPvAtt);
                $histo->set("agi_att", $gainPvAtt);
            }else{
                $attaquant->set("pv", 100);
            }
        }
    }
}else if($resultat === "defaite"){
    // si resultat = defaite
    $histo->set("result", "defaite");
    // att perd 1pv
    if($pvAtt > 0){
        $histo->set("pv_att", -1);
        $newPvAtt = $pvAtt - 1;
        $attaquant->set("pv", $newPvAtt);
        if($newPvAtt == 0){
            $attaquant->set("pos", 11);
            $histo->set("pv_att", 0);
        }
    }
}



$attaquant->update();
```

```php
$defenseur->update();
$histo->insert();

// retour
echo "une attaque à eu lieu";
```

Connecter.php

```php
<?php

// controleur : connecte l'utilisateur au jeu, verifie que les donnée
existe dans lea bdd et le connecte le cas echéant
// parametre : $_post[pseudo, mdp] - info du perso a connecter

// initialisation
require_once "utils/init.php";


$pseudo = isset($_POST["pseudo"]) ? $_POST["pseudo"] : "";
$mdp = isset($_POST["mdp"]) ? $_POST["mdp"] : "";

// traitement
$personnage = new personnage();
$idPerso = $personnage->connexion($pseudo, $mdp);

// si l'id retourné est 0 on réaffiche page co avec msg erreur
if($idPerso === 0){
    header("Location: afficher_jeu.php?error=1");
    exit;
}

// si l'id retourné est > 0 on connecte l'utilisateur et on le redirige
vers la page jeu

session_connect($idPerso);

$perso = new personnage($idPerso);
$_SESSION["last-co"] = $perso->get("date");
$date = date("Y-m-d H:i:s");
$perso->set("date", $date);
```

```php
$perso->update();


header("Location: afficher_jeu.php");
```

Creer.php

```php
<?php

// controleur : Crée un personnage dans la base de donnée
// parametre : $_POST[] : pseudo mdp for agi res - du personnage à crée

// initialistaion
require_once "utils/init.php";


// récupération des param
$pseudo = isset($_POST["pseudo"]) && $_POST["pseudo"] !== "" ?
$_POST["pseudo"] : "";
$emogi_select = isset($_POST["emogi"]) && $_POST["emogi"] !== "" ?
$_POST["emogi"] : "";
$mdp = isset($_POST["mdp_creation"]) && $_POST["mdp_creation"] !== "" ?
$_POST["mdp_creation"] : "";
$mdpVerif = isset($_POST["mdpVerif_creation"]) &&
$_POST["mdpVerif_creation"] !== "" ? $_POST["mdpVerif_creation"] : "";
$for = isset($_POST["for"]) ? $_POST["for"] : 5;
$agi = isset($_POST["agi"]) ? $_POST["agi"] : 5;
$res = isset($_POST["res"]) ? $_POST["res"] : 5;
$carac = $for + $agi + $res;

$perso = new personnage();
$pseudoVerif = $perso->verifPseudo($pseudo);


// si un champ est mal rempli on réaffiche le formulaire
if($pseudoVerif === false){
    if($mdp === "" || $mdpVerif === "" || $mdpVerif !== $mdp || $carac !==
15){
```

```
        header("Location:
afficher_form_creation.php?verif=1&error=1&for=$for&agi=$agi&res=$res&pseu
do=$pseudo&emogi=$emogi_select&mdp=$mdp&mdpVerif=$mdpVerif");
        exit;
    }else{
        header("Location:
afficher_form_creation.php?verif=1&for=$for&agi=$agi&res=$res&pseudo=$pseu
do&emogi=$emogi_select&mdp=$mdp&mdpVerif=$mdpVerif");
        exit;
    }
}else if($pseudo === "" || $mdp === "" || $mdpVerif === "" || $mdpVerif
!== $mdp || $carac !== 15){
    header("Location:
afficher_form_creation.php?error=1&for=$for&agi=$agi&res=$res&pseudo=$pseu
do&emogi=$emogi_select&mdp=$mdp&mdpVerif=$mdpVerif");
    exit;
}


//traitement
$personnage = new personnage();
$personnage->set("pseudo", $pseudo);
$personnage->set("emogi", $emogi_select);
$personnage->setMdp($mdp);
$personnage->set("for", $for);
$personnage->set("agi", $agi);
$personnage->set("res", $res);
$personnage->set("pv", 100);
$personnage->set("pos", 0);
$date = date("Y-m-d H:i:s");
$personnage->set("date", $date);

$personnage->insert();
session_connect($personnage->id());

$histo = new historique();
$histo->set("attaquant", $personnage->id());
$histo->set("action", "creation");
$histo->set("date", $date);
```

```php
$histo->insert();

$_SESSION["last-co"] = $date;

// redirection sur l'écran de jeu
header("Location: afficher_jeu.php");
```

Deconnecter.php

```php
<?php

// controleur : deconnecte le joueur


//initialisation
require_once "utils/init.php";

// traitement
session_deconnect();

// affichage
header("Location: afficher_jeu.php");
```

forToRes.php

```php
<?php

// controleur ajax : modif attribut ajoute 1 res et enleve 1 for ainsi que
3 agi
// parametre : utilisateur connecté
// retour jso des nouvelles carac

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;

// traitement
$perso = new personnage($idUser);
$for = $perso->get("for");
```

```php
$agi = $perso->get("agi");
$res = $perso->get("res");

$newFor = $for -1;
$newAgi = $agi -3;
$newRes = $res +1;

$perso->set("for", $newFor);
$perso->set("agi", $newAgi);
$perso->set("res", $newRes);

$perso->update();

$histo = new historique();
$histo->set("attaquant", $idUser);
$histo->set("action", "attribut");
$histo->set("result", "forToRes");
$date = date("Y-m-d H:i:s");
$histo->set("date", $date);

$histo->insert();

// retour
echo json_encode(["for" => $newFor, "agi" => $newAgi, "res" => $newRes]);
```

Precedente.php

```php
<?php

// controleur ajax : modif piece et consomme autant recupere autant de pv
que la nouvelle piece
// parametre : utilisateur connecté
// retour json de la nouvelle piece et des piee a afficher dans bouton

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;

// traitement
```

```php
$perso = new personnage($idUser);
$pos = $perso->get("pos");
$pv = $perso->get("pv");

$piece = $pos -1;
if($piece == 0){
    $piece = 1;
}
$preced = $piece -1;
$suiv = $piece +1;
$newPv = $pv + $piece;
if($newPv > 100){
    $newPv = 100;
}

$perso->set("pos", $piece);
$perso->set("pv", $newPv);

$perso->update();

$histo = new historique();
$histo->set("attaquant", $idUser);
$histo->set("action", "mouvement");
$histo->set("result", "precedente");
$histo->set("detail", $piece);
$date = date("Y-m-d H:i:s");
$histo->set("date", $date);

$histo->insert();


// retour
echo json_encode(["pv" => $newPv, "piece" => $piece, "preced" => $preced,
"suiv" => $suiv]);
```

Rester_inactif.php

```php
<?php

// controleur ajax surveillant l'inactiviter pour ajouter 1 point d'agi
toute les 3s.
```

```php
// parametre : $iduser - id de l'utilisateur connecté
//             $pos - position du joueur
//             $date - date en direct
//             $lastDate - date de la derniere attaque du joueur

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;

$perso = new personnage($idUser);
$pos = $perso->get("pos");
$agi = $perso->get("agi");
$date = time();

if(isset($_SESSION["date-agi-attrib"])){
    $lastTime = $_SESSION["date-agi-attrib"];
}else{
    $histo = new historique();
    $dateHisto = $histo->lastAttack($idUser);
    $dateString = $dateHisto["date"];
    $lastTime = strtotime($dateString);
}
$ecart = $date - $lastTime;

// traitement

// si on est dans les piece 1 à 9, que l'agi est <15, et que l'éart entre
lastDate et $date est > à 3s
if($agi < 15 && $pos > 0 && $pos < 10 && $ecart >= 3){
    // alors j'ajoute 1 agi par 3s d'écart et si agi dépasse 15 je fixe a
15 puis je charge dans la bdd
    $newAgi = $agi + ($ecart / 3);
    if($newAgi > 15){
        $newAgi = 15;
    }
    $perso->set("agi", $newAgi);
    $perso->update();
    $_SESSION["date-agi-attrib"] = time();
```

```
}else{
    $newAgi = $agi;
}

// retour
echo json_encode(["agi" => $newAgi]);
```

resToFor.php

```
<?php

// controleur ajax : modif attribut ajoute 1 for et enleve 1 res ainsi que
3 agi
// parametre : utilisateur connecté
// retour jso des nouvelles carac

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;

// traitement
$perso = new personnage($idUser);
$res = $perso->get("res");
$agi = $perso->get("agi");
$for = $perso->get("for");

$newRes = $res -1;
$newAgi = $agi -3;
$newFor = $for +1;

$perso->set("res", $newRes);
$perso->set("agi", $newAgi);
$perso->set("for", $newFor);

$perso->update();

$histo = new historique();
$histo->set("attaquant", $idUser);
$histo->set("action", "attribut");
```

```php
$histo->set("result", "resToFor");
$date = date("Y-m-d H:i:s");
$histo->set("date", $date);

$histo->insert();

// retour
echo json_encode(["res" => $newRes, "agi" => $newAgi, "for" => $newFor]);
```

Suivante.php

```php
<?php

// controleur ajax : modif piece et consomme autant d'agi que le nombre de
la nvlle piece
// parametre : utilisateur connecté
// retour json de la nouvelle piece et des piee a afficher dans bouton

// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;

// traitement
$perso = new personnage($idUser);
$pos = $perso->get("pos");
$agi = $perso->get("agi");

$piece = $pos +1;
$preced = $piece -1;
$suiv = $piece +1;
$newAgi = $agi - $piece;

$perso->set("pos", $piece);
$perso->set("agi", $newAgi);

$perso->update();

$histo = new historique();
$histo->set("attaquant", $idUser);
```

```php
$histo->set("action", "mouvement");
$histo->set("result", "suivante");
$histo->set("detail", $piece);
$date = date("Y-m-d H:i:s");
$histo->set("date", $date);

$histo->insert();


// retour
echo json_encode(["agi" => $newAgi, "piece" => $piece, "preced" =>
$preced, "suiv" => $suiv]);
```

Surveiller_adversaire.php

```php
<?php

// controleur ajax : récupère les adversaires présent dans la meme salle
que l'utilisateur
// parametre : $idUser - id de l'utilisateur connecté
// retour : json $adversaires - tableau d'objet perso indexé par l'id

// initialisation
require_once "utils/init.php";

include "modeles/tableaux.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;
$perso = new personnage($idUser);

// retour
include "templates/fragments/adversaires.php";
```

Surveiller_caracteristiques.php

```php
<?php

// controleur ajax : récupère les caractéristique du personnage et active
ou desactive les bouton action en fonction
// parametres : $idUser - id de l'utilisateur connecté
// retour : fichier json : pv - for - res - agi - btnactif/inactif
```

```php
// initialisation
require_once "utils/init.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;
$perso = new personnage($idUser);
$pos = $perso->get("pos");

// traitement
$suiv = $pos + 1;
// si on a plus d'agi que le numero de la piece suivante, le btn est actif
if($perso->get("agi") >= $suiv && $pos < 10){
    $btnSuiv = 1;
}else{
    $btnSuiv = 0;
}
// condition pour activer le btn preced
if($pos > 1 && $pos < 10){
    $btnPreced = 1;
}else{
    $btnPreced = 0;
}
// condition pour activé btn forToRes
if($perso->get("agi") >= 3 && $perso->get("for") > 0 && $perso->get("res")
< 15){
    $btnForToRes = 1;
}else{
    $btnForToRes = 0;
}
// condition pour activé btn resToFor
if($perso->get("agi") >= 3 && $perso->get("res") > 0 && $perso->get("for")
< 15){
    $btnResToFor = 1;
}else{
    $btnResToFor = 0;
}


// renvoi
```

```php
header('Content-Type: application/json; charset=utf-8');

$data = ["pv" => $perso->get("pv"), "for" => $perso->get("for"), "res" =>
$perso->get("res"), "agi" => $perso->get("agi"), "btnSuiv" => $btnSuiv,
"btnPreced" => $btnPreced, "btnForToRes" => $btnForToRes, "btnResToFor" =>
$btnResToFor];

echo json_encode($data);
```

Surveiller_historique.php

```php
<?php

// controleur ajax : récupère l'historique relatif à l'utilisateur
connecté
//parametres : $idUser - id de l'utilisateur connecté
//retour : fragment histo

require_once "utils/init.php";

include "modeles/tableaux.php";

// récupération
$idUser = session_isconnected() ? session_idconnected() : 0;
$perso = new personnage($idUser);

$historique = new historique();
$histos = $historique->histoPerso($idUser, $_SESSION["last-co"]);

// retour
include "templates/fragments/histo.php";
```