Projet : ticketing
Cahier des charges
Application de ticketing
Préambule
Une application de ticketing est une application permettant de suivre des demandes d'assistance, de
dépannage, des incidents, etc.…
Un ticket est donc une demande ou un incident, et l'application permet (selon les besoins), de tracer les
échanges, de suivre l'état des demandes, de trier les demandes pour gérer les priorités, de mesurer les
temps passés, etc.
Cahier des charges du client
Nous sommes le service d'assistance client d'un distributeur de produits technologiques.
Nous souhaitons mettre en place une application web de ticketing, en mode « extranet », pour permettre à
nos clients de faire leurs demandes d'assistance en ligne et de les suivre.
Nous gérons les produits par ailleurs (la gestion des produits n'est pas à réaliser ici). Nous fournirons une
table comportant : la référence et la désignation, et un champ booléen indiquant si le produit est encore en
vente ou non.
Lors de l'achat d'un produit, un compte sera ouvert pour le client si ce n'est pas déjà fait, et le produit
acheté sera enregistré (cette opération est faite par le vendeur).
Nous souhaitons que les vendeurs aient accès à une interface leur permettant :
de sélectionner ou créer un compte client
d'enregistrer une vente de produit
Pour les clients, nous ne gérons que le nom, le prénom, l'adresse de courriel (qui sert de code de
connexion), et un mot de passe (que le vendeur fera choisir au client).
L'enregistrement d'une vente consiste à effectuer les opérations suivantes :
choisir le client (recherche par l'adresse de messagerie, le nom, le prénom)
choisir le produit,
indiquer le numéro de série du produit (15 caractères, chiffres et lettres majuscules, la cohérence
doit être vérifiée, et les minuscules converties en majuscule)
vérifier la date de vente (par défaut la date du jour, mais nous souhaitons pourvoir la modifier)
Nous souhaitons une ergonomie très fluide : pour la recherche des produits, il faut que dès que l'on a tapé
au moins trois lettres, une recherche se fasse dans la base de données complète (on peut taper une partie
de la référence ou de la désignation), et si il y a moins de 10 choix possibles, ils sont proposés et

directement cliquables (et sinon, on attend plus de caractères pour proposer le choix).
Les demandes d'assistance ou de dépannage sont gérées via des tickets, créés par le client.
Le processus de traitement d'un ticket est très simple :
le client se connecte à son compte et créée un nouveau ticket (le vocabulaire à utiliser est « demande d'assistance »)
il doit choisir le produit concerné (parmi les produits qu'il a acheté, bien sûr) et décrire sa demande
le ticket est alors au statut « ouvert »
un technicien peut répondre au ticket, ou demander une précision. Le ticket passe alors à l'état « en cours de traitement ».
lorsque la demande est satisfaite, le client ou le technicien le passent à l'état résolu
Les vendeurs peuvent :
modifier leur propre mot de passe
créer un client
enregistrer une vente pour un client
Les clients peuvent :
changer leur mot de passe
changer leur adresse de messagerie
ouvrir un nouveau ticket
voir tous les tickets les concernant et leur suivi
voir les tickets en attente de réponse de leur part (c'est à dire en cours de traitement et dont le message est envoyé par le technicien)
répondre à un ticket en cours, ou apporter une précision sur un ticket ouvert ou en cours
fermer un ticket ouvert ou en cours
Les techniciens peuvent :
changer leur mot de passe
voir les tickets ouverts
voir les tickets en cours de traitement
répondre à un ticket
fermer un ticket
Les tickets ne sont pas affectés à un technicien précis (mais pour chaque réponse, on doit savoir quel
technicien l'a apportée). Nous souhaitons que les vendeurs puisse utiliser l'interface technicien, et
réciproquement.
N'hésitez pas à nous apporter des suggestions pour faciliter le traitement et la recherche des tickets (dans
le respect du processus que nous avons mis en place).

Css
Styles.css

```css
*{
    margin: 0;
    padding: 0;
```

```
    box-sizing: border-box;
}

.flex{
    display: flex;
    flex-wrap: wrap;
}
.d-none{display: none;}
.block{display: block;}
.j-center{justify-content: center;}
.j-between{justify-content: space-between;}
.j-around{justify-content: space-around;}
.j-end{justify-content: end;}
.a-center{align-items: center;}
.txt-center{text-align: center;}

body, #containConfirm{position: relative;}
.fond{
    position: fixed;
    top: 0;
    left: 0;
    z-index: -1;
    height: 100vh;
    width: 100%;
    background-image: linear-gradient(rgba(255, 255, 255, 0.8), rgba(255,
255, 255, 0.8)), url("../assets/fond_smart_DistriTech.jpg");
    background-size: cover;
    background-repeat: no-repeat;
}
#menu, #popModif{
    width: 300px;
    position: absolute;
    z-index: 1;
    top: 120px;
}
#closeMenu{
    position: relative;
    top: -24px;
    left: 150px;
}
```

```css
#confirm{
    width: 300px;
    position: absolute;
    z-index: 1;
    top: -65px;
    left: -245px;
}

.blue{color: rgb(19, 51, 92);}
.red{color: rgb(90, 24, 24);}
.white{color: white;}

.fs10{font-size: 10px;}
.fs12{font-size: 12px;}
.fs14{font-size: 14px;}
.fs16{font-size: 16px;}
.fs18{font-size: 18px;}
.fs20{font-size: 20px;}
.fs32{font-size: 32px;}

.wfit{width: fit-content;}
.w30{width: 30px;}
.w100p{width: 100%;}
.w150{width: 150px;}
.w200{width: 200px;}
.w300{width: 300px;}

.h60{height: 60px;}

.mt4{margin-top: 4px;}
.mt16{margin-top: 16px;}
.mb16{margin-bottom: 16px;}
.mt40{margin-top: 40px;}
.mrlauto{
    margin-right: auto;
    margin-left: auto;
}
.ml32{margin-left: 32px;}
.gap16{gap: 16px;}
.titre{height: 80px;}
```

```css
table{border-spacing: 10px;}
tr{position: relative;}
.tdPop{
    z-index: 1;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 5px;
    padding: 8px 16px;
}

.logo{
    height: 60px;
    width: 60px;
    background-image: url("../assets/logoTechoTb.jpg");
    background-size: cover;
    border-radius: 5px;
}

.encart{
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 5px;
    padding: 32px 0 32px 0;
}

ul li {list-style: none;}

input {
    padding: 4px 8px;
}
button, input[type="submit"] {
    color: white;
    background-color: rgb(32, 76, 134);
    border: none;
    border-radius: 3px;
}
.btn{
    background-color: rgb(32, 76, 134);
    border-radius: 3px;
```

```
}
.btnPad{padding: 2px 4px;}
:visited, :active, :link{
    text-decoration: none;
    color: rgb(19, 51, 92);
}
```

Javascript

client.js

```
let mdpClient = document.getElementById("mdpClient");
let btnMdpClient = document.getElementById("btnMdpClient");
btnMdpClient.addEventListener("click", (e) =>{
    if(mdpClient.type === "password"){
        mdpClient.type = "text";
    }else{
        mdpClient.type = "password";
    }
})
```

connexion.js

```
let mdp = document.getElementById("mdp");
let btnMdp = document.getElementById("btnMdp");
btnMdp.addEventListener("click", (e) =>{
    if(mdp.type === "password"){
        mdp.type = "text";
    }else{
        mdp.type = "password";
    }
})
```

historique.js

```
// je récupère tout les tr du document
let histos = document.querySelectorAll(".histo");

document.addEventListener("click", (e) => {
    histos.forEach(histo => {
        let tdPop = histo.querySelector(".tdPop");
        let tdClose = histo.querySelector(".tdClose");
```

```
        if(tdPop.classList.contains("d-none") &&
histo.contains(e.target)){
            tdPop.classList.remove("d-none")
        }else if(!tdPop.classList.contains("d-none") &&
tdClose.contains(e.target)){
            tdPop.classList.add("d-none");
        }
    })
})
```

modif.js

```
let popModif = document.getElementById("popModif");
function cachePop(){
    popModif.classList.add("d-none");
}
setInterval(cachePop, 2000);
```

modification.js

```
let mdpOld = document.getElementById("mdpOld");
let btnMdpOld = document.getElementById("btnMdpOld");
btnMdpOld.addEventListener("click", (e) =>{
    if(mdpOld.type === "password"){
        mdpOld.type = "text";
    }else{
        mdpOld.type = "password";
    }
})

let mdpNew = document.getElementById("mdpNew");
let btnMdpNew = document.getElementById("btnMdpNew");
btnMdpNew.addEventListener("click", (e) =>{
    if(mdpNew.type === "password"){
        mdpNew.type = "text";
    }else{
        mdpNew.type = "password";
    }
})

let mdpVerif = document.getElementById("mdpVerif");
let btnMdpVerif = document.getElementById("btnMdpVerif");
```

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

msg.js

```
let cloture = document.getElementById("cloture");
let confirme = document.getElementById("confirm");
let annule = document.getElementById("annule");

cloture.addEventListener("click", (e) => {
    confirme.classList.remove("d-none");
})

annule.addEventListener("click", (e) => {
    confirme.classList.add("d-none");
})

// appel le controleur ajax de surveillance des message
let listMsg = document.getElementById("listMsg");


function surveilleMsg(){
    fetch("surveiller_msg.php")
    .then(resp => {
        return resp.text();
    })
    .then(retour => {
        listMsg.innerHTML = retour;
    })
}
surveilleMsg();
setInterval(surveilleMsg, 3000);
```

param.js

```
let param = document.getElementById("param");
let menu = document.getElementById("menu");
```

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

param.addEventListener("click", (e) =>{
    menu.classList.remove("d-none");
})

closeMenu.addEventListener("click", (e) => {
    menu.classList.add("d-none");
})
```

ticket.js

```
let param = document.getElementById("param");
let menu = document.getElementById("menu");
let closeMenu = document.getElementById("closeMenu");

param.addEventListener("click", (e) =>{
    menu.classList.remove("d-none");
})

closeMenu.addEventListener("click", (e) => {
    menu.classList.add("d-none");
})
```

Utils

Field.php

```
<?php

// classe permetant de gérer un champ de base de données

class _field {

    protected $name;         // nom du champ
    protected $type;         // type de champ : TXT, DATE, DATETIME, NUM,
LINK
    protected $libelle;      // Libellé du champ
    protected $link;         // Objet pointé si lien

    protected $object;       // Objet dont ce champ fait partie

    protected $value;        // valeur de l'objet
```

```
    protected $target;      // Objet pointé si chargé (non maitrisé à ce
stade)


    function __construct($object, $name, $type = null, $libelle  = null,
$link = null) {
        // Paramètres :
        //      $object:    objet de ratachement
        //      $name;        // nom du champ
        //      $type;        // type de champ : TXT, DATE, DATETIME, NUM,
LINK - par defaut : TXT
        //      $libelle;     // Libellé du champ - par défaut : nom du
champ
        //      $link;        // Objet pointé si lien (facultatif) - si
c'est un lien et que link n'est pas précisé,link = name

        $this->object = $object;
        $this->name = $name;
        $this->type = empty($type) ? "TXT" : $type;
        $this->libelle = empty($libelle) ? $name : $libelle;
        if ($type == "LINK") $this->link = empty($link) ? $name : $link;
        else $this->link = $name;
    }

    function get() {
        // Role: récupérer la valeur d'un champ
        // Paramètres : aucun
        // Retour : la valeur du champ

        return $this->value;
    }

    function set($value) {
        // Role: charger la valeur du champ
        // Paramètres : $value : la valeur à charger
        // Retour : true false

        $this->value = $value;
```

```
        return true;
    }

    function html() {
        // Role: récupérer la valeur HTML du champ
        // Paramètres : aucun
        // Retour : la valeur du champ avec la gestion des balise html

        return nl2br(htmlentities($this->get()));
    }

    function type() {
        // Role: récupérer le type du champ
        // Paramètres : aucun
        // Retour : le code du type

        return $this->type;
    }

    function libelle($html = true) {
        // Role: récupérer le libelle du champ
        // Paramètres : $html - true si on veut le convertir en HTML -
true par défaut
        // Retour : le code du type

        if($html) return nl2br(htmlentities($this->libelle));
        else return $this->libelle;
    }

    function link(){
        // Role: récupérer la valeur du lien d'un champ
        // Paramètres : aucun
        // Retour : la valeur du lien du champ

        return $this->link;
    }

}
```

Init.php

```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/field.php";
include "utils/model.php";
include "utils/user.php";

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

// propriété de debug
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

// chargement des librairies

include "modeles/utilisateur.php";
include "modeles/produit.php";
include "modeles/vente.php";
include "modeles/ticket.php";
include "modeles/message.php";
include "utils/session.php";


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

Model.php

```php
<?php

/*
```

```
Classe _model : modèle générique de gestion des objets // avec la gestion des champs par l'objet

Utilisation :

    * Ouverture bdd :
        bdd() : crée la chaine de connexion à la base de donnée si elle n'existe pas et la retourne

    * Méthodes protégées (utilisable dans les classes filles) :
        define() : protected - a implémenter dans la classe fille en utilisant addfield à l'interieur
        get_$field() : vérification directement dans get($field), si un get spé existe on lance cette méthode, sinon lance la méthode générique
        verify() : vérifie la cohérence d'un objet

    * Méthodes neutres :
        __construct($id) : chargement d'un nouvel objet via l'id en une seule fois
        is() : vérifie si l'objet est chargé ou non

    * Méthodes magiques :
        __get() : utilisé lorsque l'on fait $objet->attribut ($name sera donc à la place de "attribut")
        __set() : utilisé lorsque l'on fait $objet->attribut = valeur (attribut sera $name et valeur $value)

    * Getters :
        id() : récupère l'id de l'objet
        get($field) : récupère la valeur stocké dans le champ ciblé de l'objet
        getHTML($field) : retourne le champ formaté de façon à transformer les ballises html en texte
        getTarget($field) : récupère l'objet de la classe pointé par le champ

    * Setters :
        set($field, $value) :  charge le champ d'un objet avec sa valeur
        loadFromTab($table) : charge un objet avec un tableau de donnée (récupéré par fetch)
```

```
    * Méthodes de synchronisation avec la bdd :
        advancedLoad($param) :  : charge de façon autonome l'objet courant
        load($id) : chargement de l'objet courant par l'id
        insert() : insert un objet dans la base de donnée
        update() : met à jour un objet existant dans la base de donnée
        delete() : supprime un objet dans la base de donnée
        listAll($filtre, $tri, $limit) : récupère la liste de tout les
objet de la table et si ils sont spécifié y inclus des condition de
récupération et de triage

    * Méthodes permettant de gérer les champs (création d'un objet champ)
:
        addfield() : permet d'ajouter un champ à l'objet courant en
utilisant l'objet champs
        getField(): récupère l'objet correspondant çà un champ
        getAllFields() : récupère tout les champs d'un objet

    * Sous méthodes :
        toTab() : récupère les champs et valeurs d'un objet et les
transforme en tableau
        makeSet() : construit la partie SET d'une requete insert ou update
        makeParam() : construit les parametre nécessaire à insert et
update
        makeFilter($filter) : construit la requete à mettre derriere le
WHERE
        makeParamFilter($filter) : construit le tableau de parametre en
fonction des filtres demandé
        makeTri($tri) : construit la requete à mettre derriere le ORDER BY
        makeFields() : consrtuit la liste des champs à mettre dans un
SELECT (id + tt les champs d'une classe)
        runSql($sql, $param=NULL) : prépar et execute la requete sql
        recoverReqSimple($req) : récupère le résultat de la requete
(lorsque le résultat attendu est unique)
        recoverReqMulti($req) : récupère le résultat de la requete
(lorsque le résultat attendu est multiple)

*/

class _model {
```

```php
    // Attributs
        protected $table = "";  // Table :
        protected $fields = []; // liste des champs de la classe


    // stockage
        protected $id = 0;  // stockage de l'id
        protected $targets = [];    // objet chargé des liens récupéré
["champ" => objetLié, ...]


    // Ouverture bdd :

        // création de la variable statique
        protected static $bdd;

        // ouverture
        static function bdd() {
            // Rôle : crée la chaine de connexion à la base de donnée si
elle n'existe pas et la retourne
            // Paramètres : aucun
            // Retour : $bdd - chargé avec la chaine de connexion a la bdd
(objet PDO)

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


    // Méthodes //

        // Méthodes protégées (utilisable dans les classes filles) :

            protected function define() {
```

```
                // Rôle : protected - a implémenter dans la classe fille
en utilisant addfield à l'interieur pour définir les champs de la classe
                // Paramètres : aucun
                // Retour : aucun
            }

            function verify(){
                // role : vérifie la cohérence d'un objet
                // parametre : aucun
                // retour : true si cohérent sinon false

                return true;
            }



        // Neutres :

            function __construct($id = NULL){
                // cette fonction ce déclenche à chaque instanciation
d'une classe, le parametre devra donc etre mis dans les parenthèse lors de
cette instanciation
                // role : instancie un nouvel objet par l'id
                // parametre : $id - id de l'objet à instancier
                // retour: constructeur dc pas de retour

                $this->define();

                if(!is_null($id)) $this->load($id);

            }

            function is(){
                // role : test si un objet est chargé
                // parametre : aucun
                // retour : true / false

                return !empty($this->id);
            }
```

```
        // Méthodes magiques :

            function __get($name){
                // role : utilisé lorsque l'on fait $objet->attribut
($name sera donc à la place de "attribut")
                // parametres : $name - le champs visé
                // retour : la valeur que l'on veut récupérer

                if($name === "id") return $this->id();
                else if(in_array($name, $this->fields)) return
$this->get($name);

            }

            function __set($name, $value){
                // role : utilisé lorsque l'on fait $objet->attribut =
valeur (attribut sera $name et valeur $value)
                // parametre : $name - champs que l'on souhaite charger
                //             $value - valeur à charger dans le champ
                // retour : aucun

                if(in_array($name, $this->fields)) $this->set($name,
$value);
            }


        // Getters :

            function id(){
                // role : retourne l'id de l'objet courant
                // paramere : aucun
                // retour : $id - id de l'objet courant

                return $this->id;
            }

            function get($field){
                // role : récupère la valeur stocké dans le champs ciblé
de l'objet courant
```

```
                // parametre : $field - le champs ciblé
                // return : la valeur stocké dans le champ ciblé ou chaine
vide si

                // vérife si une méthode spécifique existe et retourne son
résultat le cas échéant
                if(method_exists($this, "get_$field")){
                    $method = "get_$field";
                    return $this->$method;
                }

                if(isset($this->fields[$field])) return
$this->fields[$field]->get();
                else return "";
            }

            function getHTML($field){
                // role : retourne le champ formaté de façon à transformer
les ballises html en texte
                // parametre : $field - le champ ciblé
                // retour : la valeur du champ tra nsformé

                return nl2br(htmlentities($this->get($field)));
            }

            function getTarget($field){
                // role : récupère l'objet de la classe pointé par le
champ
                // parametre : $field - le champs pointant l'objet à
récupérer
                // retour : si l'objet est déjà chargé, on retourne
l'objet chargé
                //          sinon, si le champ n'est pas un lien, retourne
un nouvel objet de la class model
                //                 si le champ est un lien, retourne un
objet de la class en question non chargé ou vierge si la cible est vide

                // si l'objet n'est pas déjà chargé
                if(!isset($this->targets[$field])){
                    // si le champ est un lien
```

```
                    if(isset($this->fields[$field])){
                        $class = $this->fields[$field]->link();
                        $this->targets[$field] = new
$class($this->get($field));
                    }else{
                        // sinon
                        $this->targets[$field] = new _model();
                    }
                }

                // retour
                return $this->targets[$field];

            }


        // Setters :

            function set($field, $value){
                // role : charge le champ d'un objet avec sa valeur
                // parametre : $field - champ à remplir
                //             $value - valeur à enregistrer dans le champ
                // retour : true / false

                // vérifie l'existence du champ ciblé
                if(isset($this->fields[$field]))
$this->fields[$field]->set($value);
                else return false;

                // retour
                return true;
            }

            function loadFromTab($table){
                // Role : initialise un objet avec un tableau de donnée
(récupéré par fetch)
                // paramètre : $table - le tableau de donnée récupéré
                // retour : true si ok / false sinon
                // si $table est vide je retourne false
                if(empty($table)) return false;
```

```
                // si j'ai un id dans le tableau, je le charge dans
l'objet courant
                if(isset($table["id"])) $this->id = $table["id"];

                // pour chaque champ du tableau si il y a une valeur, je
le charge dans le champ de l'objet courant
                foreach($this->fields as $field => $objet){
                    if(isset($table[$field]))
$this->fields[$field]->set($table[$field]);
                }

                // retour
                return true;
            }


        // Méthodes de synchronisation avec la bdd :

            function advancedLoad($param){
                // role : charge de façon autonome l'objet courant
                // parametre : $param - si c'est un id : charge l'objet à
partir de l'id
                //                    - si c'est un tableau, charge
l'objet courant avec les champs=>valeur du tableau
                //                    - si c'est un objet, charge l'objet
courant avec cet objet
                // retour : true / false si ça a fonctionné

                // on test le parametre
                if(is_numeric($param)) $this->load($param);
                else if(is_array($param)) $this->loadFromTab($param);
                else if($param instanceof _model)
$this->loadFromTab($param->toTab());

                return $this->is();
            }

            function load($id){
            // role : chargement de l'objet courant par l'id
```

```
            // parametre : $id - id de l'objet à récupérer
            // retour : true si ok / false sinon

                // construction
                $sql = "SELECT `id`, ". $this->makeFields() . " FROM
`$this->table` WHERE `id` = :id";
                $param = [":id" => $id];

                // préparation/execution
                $req = $this->runSql($sql, $param);

                // récupération / retour
                return $this->recoverReqSimple($req);
            }

            function insert(){
                // role : insert un objet dans la base de donnée
                // parametres : aucun
                // retour : true false

                // construction
                $sql = "INSERT INTO `$this->table` SET " .
$this->makeSet();
                $param = $this->makeParam();

                // préparation/éxécution
                $this->runSql($sql, $param);

                // récupération de l'id
                $bdd = static::bdd();
                $this->id = $bdd->lastInsertId();

                //retour
                return true;
            }

            function update(){
                // role : met à jour un objet existant dans la base de
donnée
                // parametres : aucun
```

```
                // retour : true false

                // construction
                $sql = "UPDATE `$this->table` SET " . $this->makeSet() . "
WHERE `id` = :id";
                $param[":id"] = $this->id();
                $param += $this->makeParam();
                print_r($this->id());
                print_r($this->makeSet());
                print_r($this->makeParam());
                // préparation/éxécution
                $this->runSql($sql, $param);

                //retour
                return true;
            }

            function delete(){
                // role : supprime un objet dans la base de donnée
                // parametres : aucun
                // retour : true false

                // construction
                $sql = "DELETE FROM `$this->table` WHERE `id` = :id";
                $param = [":id" => $this->id];

                // préparation/récupération
                $this->runSql($sql, $param);

                // retour
                return true;
            }

            function listAll($filter = [], $tri = [], $limit = NULL){
                // role : récupère la liste de tout les objet de la table
et si ils sont spécifié y inclus des condition de récupération et de
triage
                // parametres : $filter - tableau indexé par le champ ex
["champnom" => "valeurnom", etc..]
```

```
                //              $ tri - tableau simple avec - ou + pour
asc desc et le champ qui sert de tri - ex ["-nom", etc..]
                //              $limit - nbre entier qui fixe la limite à
récupérer
                // retour : un tableau d'objet indexé par l'id

                // construction
                $sql = "SELECT `id`, " . $this->makeFields() . " FROM
`$this->table`";
                $param = [];
                if(!empty($filter)){
                    $sql .= " WHERE " . $this->makeFilter($filter);
                    $param = $this->makeParamFilter($filter);
                }
                if(!empty($tri)) $sql .= " ORDER BY " .
$this->makeTri($tri);
                if(!is_null($limit)) $sql .= " LIMIT $limit";

                // préparation/éxecution
                $req = $this->runSql($sql, $param);

                // récupération/retour
                return $this->recoverReqMulti($req);
            }


        // Méthodes permettant de gérer les champs (création d'un objet
champ) :

            protected function addField($name, $type = NULL, $libelle =
NULL, $link = NULL){
                // permet d'ajouter un champ à l'objet courant en
utilisant l'objet champs
                // parametres : $name - Nom du champ
                //              $type - facultatif - type de champ
(TXT-DATE-LINK)
                //              $libelle - facultatif - libellé du champ
                //              $link - facultatif - objet pointé par le
champ
```

```
                $this->fields[$name] = new _field($this, $name, $type,
$libelle, $link);
            }

            function getField($field){
                // role : récupère l'objet correspondant à un champ
                // parametre : $field - le champ recherché
                // retour : l'objet champ si existe sinon un objet champ
vierge avec un nom _

                if(isset($this->fields[$field])) return
$this->fields[$field];
                else return new _field($this, "_");
            }

            function getAllFields(){
                // role : récupère tout les champs d'un objet
                // parametres : aucun
                // retour : un tableau simple des champs de l'objet

                return $this->fields;
            }


        // Sous méthodes :

            function toTab(){
                // Role : récupèreles champs et valeurs d'un objet et les
transforme en tableau
                // parametres : aucun
                // retour : tableau de valeur indexé par le nom du champs

                // initialisation du tableau
                $tab = [];
                // pour chaque champ
                foreach($this->fields as $field => $objet){
                    // récupération de la valeur qu'on rajoute à l'indexe
champ du tableau
                    $tab[$field] = $this->fields[$field]->get();
                }
```

```
                // retour du tableau
                return $tab;
            }

            function makeSet(){
                // role : construit la partie SET d'une requete insert ou
update
                // parametre : aucun
                // retour : une chaine de caractère chargé avec la requete
construite

                // construction d'un tableau construit avec le nom du
champ = :nomduchamp, pour chaque champ
                $tab = [];
                foreach($this->fields as $field => $objet){
                    $tab[] = "`$field` = :$field";
                }
                // transformation du tableau en chaine caractère avec
chaque element du tab séparé par ", " et retour
                return implode(", ", $tab);
            }

            function makeParam(){
                // role : construit les parametre nécessaire à insert et
update
                // parametre : aucun
                // retour : un tableau indexé par le parametre - ex :
[":nomchamp" => valeurchamp]

                // pour chaque champ, construction du tableau
                $tab = [];
                foreach($this->fields as $field => $objet){
                    // si le champ à une valeur stocker, je la charge
                    if(isset($this->fields[$field])) $tab[":$field"] =
$this->fields[$field]->get();
                    // sinon je charge vide
                    else $tab[":$field"] = NULL;
                }

                // retour
```

```
                return $tab;
            }

            function makeFilter($filters){
                // role : construit la requete à mettre derriere le WHERE
                // parametre : $filter - un tableau indexé par le champ à
filtrer - ex ["nomChamp" => "valeurChamp", etc...]
                // retour : la requete sous forme de chaine de caratctere
- ex "`nomchamp` = :nomchamp, etc..."

                $tab = [];
                foreach($filters as $filter => $value){
                    $tab[] = "`$filter` = :$filter";
                }

                return implode(", ", $tab);
            }

            function makeParamFilter($filters){
                // role : construit le tableau de parametre en fonction
des filtres demandé
                // role : $filter - un tableau indexé par le champ à
filtrer - ex ["nomChamp" => "valeurChamp", etc...]
                // retour : un tableau indexé par le champ - ex [":param"
=> "valeurParam", etc...]

                $tab = [];
                foreach($filters as $filter => $value){
                    $tab[":$filter"] = $value;
                }

                return $tab;
            }

            function makeTri($tris){
                // role : construit la requete à mettre derriere le ORDER
BY
                // parametre : $tri - un tableau simple avec le champ et
l'ordre - ex ["-nomChamp", etc]
                // retour : la requete sous forme de chaine de caratctere
```

```
                $tab = [];

                foreach($tris as $tri){
                    // récupère la direction +/- qui est sur le premier
carac
                    $dir = substr($tri, 0, 1);

                    if($dir === "-"){
                        // si dir = - on fixe tri descendant
                        $order = "DESC";
                        // récupère le champ qui sert de tri qui commence
au 2e caractere
                        $field = substr($tri, 1);
                    }else if($dir === "+"){
                        // si dir = + on fixe tri ascendant
                        $order = "ASC";
                        // récupère le champ qui sert de tri qui commence
au 2e caractere
                        $field = substr($tri, 1);
                    }else{
                        // si ce n'est ni + ni - c'est qu'il n'est pas
spécifié, on fixe ascebndant par défaut
                        $order = "ASC";
                        // et on récupère le champ qui commence du coup au
premier caractère
                        $field = $tri;
                    }
                    $tab[] = "`$field` $order";
                }

                // retour de la requete
                return implode(", ", $tab);

            }

            function makeFields(){
                // role : construit la liste des champs à récupérer
                // parametre : aucun
```

```
                // retour : $select - une chaine de caractère avec la
liste des champs

                // initialisation du tableau
                $array = [];

                // chargement du tableau avec les champs de l'objet
                foreach($this->fields as $field => $objet){
                    $array[] = "`$field`";
                }


                // retour
                return implode(", ", $array);

            }

            function runSql($sql, $param=[]){
            // role : construit la préparation, l'executionde la requete
sql
            // parametre : $sql - la requete elle meme
            //             $param - les parametre de la requete
            // retour : retourne la requete construite ou false si echec

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

                // retour
                return $req;
            }

            function recoverReqSimple($req){
```

```
                // Role : récupère le résultat de la requete lorsqu'il est
unique et le charge
                // parametre : $req - la requete en question
                // retour : true / false
                $result = $req->fetch(PDO::FETCH_ASSOC);
                // traitement
                // si vide : retourne false
                if(empty($result)) return false;

                //sinon : charge l'objet avec le résultat
                $this->loadFromTab($result);

                // retour
                return true;
            }

            function recoverReqMulti($req){
                // Role : récupère le résultat de la requete lorsqu'il est
multiple
                // parametre : $req - la requete en question
                // retour : tableau d'objet indexé par l'id

                // récupération
                $array = [];
                while($result = $req->fetch(PDO::FETCH_ASSOC)){
                    $class = get_class($this);
                    $objet = new $class();
                    $objet->loadFromTab($result);
                    $array[$objet->id()] = $objet;
                }

                // retour
                return $array;
            }
}
```

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

```php
    // 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 $utilisateurConnecte;
        return $utilisateurConnecte;
    } else {
        return new utilisateur();
    }


 }
```

```php
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;
    //    - charger l'objet utilisateur connecté
    global $utilisateurConnect;
    $utilisateurConnect = new utilisateur(session_idconnected());
}
```

User.php

```php
<?php

/*

classe user étendu de la classe model

    Utilisation :

        * Parametrage :
            const LOGIN = "" - champ utilisé pour stocker l'identifiant
            const PWD = "" - champ utilisé pour stockerle mot de passe

        * Méthodes :
            genPwd() : génère un mot de passe automatique
            setPwd($pwd) : crypte le mdp et le charge dans le champ "pwd"
(nom à adapter au projet - mdp, pwd, etc..) de l'objet courant (condition
: mini 8 caractere, dont 1 majuscule, 1 chiffre et 1 carctere spécial)
```

```
            sendConfirm() : envoie un mail de confirmation avec le mot de
passe auto
            loginVerify($login, $pwd) : verifie la concordance du mot de
passe saisie et enregistré (nom du champ login à adapter au projet, idem
pour le mdp)

*/

class _user extends _model {

    const LOGIN = "";
    const PWD = "";

    function genPwd(){
        // role : génère un mot de passe automatique
        // parametre : aucun
        // retour : une chaine de caractere avec le mot de passe si ok ou
vierge sinon

        // spécification des caractère, 1 majuscule, 1 chiffre, 1
caractere spé, et 5 lettre minuscule

        $majuscule = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        $special = "@[]^_!\"#$%&\'()*+,-./:;{}<>=|~?";
        $minuscule = "abcdefghijklmnopqrstuvwxyz";

        // génération
        $int = random_int(0, 8);
        $maj = $majuscule[random_int(0, strlen($majuscule)-1)];
        $spe = $special[random_int(0, strlen($special)-1)];
        $min = "";
        for($i=0; $i<5; $i++){
            $min .= $minuscule[random_int(0, strlen($minuscule)-1)];
        }

        // mélange
        $pwd = $int . $maj . $min . $spe;
        $pwd = str_shuffle($pwd);
        // retour
        return $pwd;
```

```
    }

    function setPwd($pwd){
        // role : crypte le mdp (condition : mini 8 caractere, dont 1
majuscule ([A-Z]), 1 chiffre (/\d/) et 1 carctere spécial (/[\W_]/))
        //        puis le charge à l'emplacement du mdp (nom à adapter au
projet)
        // parametre : $pwd - le mdp à crypté
        // retour : true / false

        // si le mdp ne rempli pas toute les condition, je retourne false
        if(strlen($pwd) < 8 || preg_match('/\d/', $pwd) == 0 ||
preg_match('/[A-Z]/', $pwd) == 0 || preg_match('/[\W_]/', $pwd) == 0)
return false;

        // hash le mdp et le charge dans l'objet courant
        $hash = password_hash($pwd, PASSWORD_DEFAULT);
        $this->fields[$this::PWD]->set($hash);

        // retour
        return true;
    }

    function sendConfirm($template, $param){
        // role : envoie un mail de confirmation avec le mot de passe auto
        // parametre : $template - le nom du template à inclure
        //             $param - un tableau indexé avec les infos
necessaire à l'envoi du mail ex ["nom" => valeurNom, etc...]
        // retour : send / echec

        // destinataire
        $prenom = $param["prenom"];
        $nom = $param["nom"];
        $mailTo = "alaugier@mywebecom.ovh"; // ceci est pour les test, en
condition réel, il faudra mettre $param["mail"]
        $to = "'$prenom $nom' <$mailTo>";
        // sujet
        $subject = "Confirmation de votre inscription";
        //expediteur
            // en tete
```

```php
            $head = [];
            $head["From"] = "'Ticketing' <laugierantoine@gmail.com>";
            $head["reply-to"] = "laugierantoine@gmail.com";
            // info pour création mail html
            $head["MIME-Version"] = "1.0";
            $head["Content-Type"] = "text/html; charset=UTF-8";
        // insertion template
        ob_start();
        include "$template";
        $message = ob_get_clean();
        //retour
        if(mail($to, $subject, $message, $head)) return "Send";
        else return "Echec";
    }

    // action sur la bdd
    function loginVerify($login, $pwd){
        // role : verifie la concordance du mot de passe saisie avec celui enregistré
        // parametre : $login - identifiant de l'utilisateur (nom du champ recherché à adapter au projet - nom, mail, etc..)
        //             $pwd - mdp saisi par l'utilisateur à comparer avec le champ mdp (nom à adapter au projet - mdp, pwd, etc..)
        // retour : l'id de l'utilisateur ou 0

        //construction
        $sql = "SELECT `id`, `".$this::PWD."` FROM `$this->table` WHERE `".$this::LOGIN."` = :identifiant";
        $param = [":identifiant" => $login];

        // dérouler
        $req = $this->runSql($sql, $param);

        // récupération
        $this->recoverReqSimple($req);

        if($this->is() && password_verify($pwd, $this->get($this::PWD))) return $this->id();
        else return 0;
    }
```

```
}
```

Verif_connexion.php

```php
<?php
/*

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

*/

// surveillance d'une erreur de connexion en parametre
$error = isset($_GET["error"]) ? $_GET["error"] : 0;

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

Modeles

Message.php

```php
<?php

class message extends _model {

    protected $table = "message";

    function define(){
        $this->addField("contenu");
        $this->addField("ticket", "LINK", "Ticket", "ticket");
        $this->addField("auteur", "LINK", "Auteur", "utilisateur");
        $this->addField("date");
    }
}
```

Produit.php

```php
<?php
```

```php
class produit extends _model {

    protected $table = "produit";

    function define(){
        $this->addField("ref");
        $this->addField("design");
        $this->addField("ev");
    }

    function snCorrespond($snVente){
        // role : compare les 7 premier caractères du sn avec la ref du
produit
        // parametre : $ref - ref du produit comparé et
        //             $snVente - n°série a comparé
        //retour : 1 si ok 0 sinon;

        // préparation de la chaine a comparer
        $snCut = substr($snVente, 0, 7);

        // si la ref est égal a la chaine coupé et la longueur total = à
15, je retourne 1, sinon 0
        if($snCut === $this->get("ref") && strlen($snVente) === 15){
            return 1;
        }else{
            return 0;
        }
    }

        function idRef($ref){
            // role : récupère l'id d'un produit via sa reférence
            // parametre : $ref - ref du prdouit
            // retour : $id - id du produit

            // construction
            $sql = "SELECT `id` FROM `produit` WHERE `ref` = :ref";
            $param = [":ref" => $ref];

            //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 0;
            }

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

            // retour
            return $id;

        }
}
```

Ticket.php

```
<?php

class ticket extends _model {

    protected $table = "ticket";

    function define(){
        $this->addField("client", "LINK", "Client", "utilisateur");
        $this->addField("produit", "LINK", "Produit", "produit");
        $this->addField("demande");
        $this->addField("statut");
        $this->addField("lastAut", "LINK", "Auteur de la dernière
réponse", "utilisateur");
        $this->addField("ouverture");
        $this->addField("fermeture");
```

```
        $this->addField("par", "LINK", "Cloturé par", "utilisateur");
    }

    function listePerso($id){
        // role : récupération des ticket correspondant à l'id
        // parametre : $id - id du client rattaché au ticket
        // retour : $liste - tableau d'objet ticket indexé par l'id

        // construction
        $sql = "SELECT `client`, `produit`, `statut`, `lastAut`,
`ouverture` FROM `ticket` WHERE `id` = :id ORDER BY `ouverture` DESC";
        $param = [":id" => $id];

        // 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
        $liste = [];
        while($result = $req->fetch(PDO::FETCH_ASSOC)){
            $tick = new ticket();
            $tick->loadFromTab($result);
            $liste[$tick->id()] = $tick;
        }

        // retour
        return $liste;
    }
}
```

Utilisateur.php

```
<?php
```

```php
class utilisateur extends _user {

    const LOGIN = "mail";
    const PWD = "mdp";

    protected $table = "utilisateur";

    function define(){
        $this->addField("type");
        $this->addField("nom");
        $this->addField("prenom");
        $this->addField("mail");
        $this->addField("mdp");
    }

    function testId($mail, $prenom, $nom){
        // role : récupère l'id de l'utilisateur répondant aux parametre
        // parametre : $mail, prenom et nom - de l'utilisateur recherché
        // retour : $id de l'utilisateur

        // construction
        $sql = "SELECT `id` FROM `utilisateur` WHERE `mail` = :mail AND
`prenom` = :prenom AND `nom` = :nom";
        $param = [":mail" => $mail, ":nom" => $nom, ":prenom" => $prenom];

        //préparation
        $req = $this->runSql($sql, $param);

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

        // retour
        return $id;
    }
```

```
}
```

Vente.php

```php
<?php

class vente extends _model {

    protected $table = "vente";

    function define(){
        $this->addField("vendeur", "LINK", "Vendeur", "utilisateur");
        $this->addField("client", "LINK", "Client", "utilisateur");
        $this->addField("produit", "LINK", "Produit", "produit");
        $this->addField("sn");
        $this->addField("date");
    }

}
```

Templates
Fragments
Change_espace.php

```php
<?php
// fragment d'affichage de l'espace et bouton de changement

echo "<section class='flex j-between a-center'>";
if($espace === "vendeur"){
    ?>
        <h4 class="fs20">Espace Vendeur</h4>
        <a href="changer_espace.php?espace=technicien"><button
class="btnPad fs12">Espace Technicien</button></a>
    <?php
}else if($espace === "technicien"){
    ?>
        <a href="changer_espace.php?espace=vendeur"><button class="btnPad
fs12">Espace Vendeur</button></a>
        <h4 class="fs20">Espace Technicien</h4>
    <?php
}
```

```
echo "</section>";
```

Espace_vendeur.php

```
<?php
// fragment html de l'espace vente
?>

<div class="mt40">
    <a href="afficher_client.php"><button id="client" class="btnPad">Créer
un client</button></a>

    <h5 class="fs18 mt40"></h5>
    <form class="encart mt40" action="afficher_form_vente.php"
method="POST">
        <div class="w200 mrlauto">
            <label class='w100p block fs16' for='idProduit'>Sélectionner
un produit :</label>
            <select class="w100p mt4" name='idProduit' id='idProduit'>
                <?php include
"templates/fragments/select_produit_vente.php"; ?>
            </select>
            <?php
                if($noProd == 1){
                    echo "<p class='w200 mt4 fs12 red mrlauto'>Veuillez
selectionner un produit</p>";
                }
            ?>
        </div>
        <div class="w200 mrlauto mt16 flex j-between a-center">
            <input class="fs14" type="submit" value="Enregistrer une
vente">
        </div>
    </form>
</div>
```

Header.php

```
<?php
```

```
// fragment à insérer avant le main d'un template
?>
<div class="fond"></div>
<div class="titre flex a-center j-between w300 mrlauto">
    <h1 class="fs32 blue wfit">TechoTb</h1>
    <div class="logo"></div>
</div>
```

Histo_ticket_client.php

```
<?php
// fragment : création de la liste des tickets client
?>
<thead>
    <tr>
        <th class="fs12">produit</th>
        <th class="fs12">Statut</th>
        <th class="fs12">Ouverture</th>
    </tr>
</thead>
<tbody>
<?php
foreach($liste as $id => $ticket){
    $produit = $ticket->getTarget("produit");
    ?>
    <tr class="histo">
        <td class="fs10"><?= $produit->get("ref") ?></td>
        <td class="fs10"><?= $ticket->get("statut") ?></td>
        <td class="fs10"><?= $ticket->get("ouverture") ?></td>
        <td class="w200 tdPop fs10 mrlauto txt-center d-none flex
j-between a-center gap16"><div><?= $produit->get("design") ?><br><?=
$ticket->getHTML("demande") ?><br><a
href="afficher_ticket.php?idTicket=<?= $id ?>">Détail ⇛</a>
        <?php
            if($ticket->get("fermeture") != ""){
                ?>
                <br> cloturé le <?= $ticket->get("fermeture") ?> par <?
$ticket->get("par") ?>
                <?php
            }
        ?>
```

```
        </div><button class="tdClose btnPad">✖</button></td>
    </tr>
    <?php
}
?>
</tbody>
```

Histo_ticket.php

```
<?php
// fragment : création de la liste des tickets
?>
<thead>
    <tr>
        <th class="fs12">Client</th>
        <th class="fs12">Produit</th>
        <th class="fs12">Statut</th>
        <th class="fs12">Ouverture</th>
    </tr>
</thead>
<tbody>
<?php
foreach($liste as $id => $ticket){
    $client = $ticket->getTarget("client");
    $produit = $ticket->getTarget("produit");
    ?>
    <tr class="histo">
        <td class="fs10"><?= $client->getHTML("nom") ?></td>
        <td class="fs10"><?= $produit->get("ref") ?></td>
        <td class="fs10"><?= $ticket->get("statut") ?></td>
        <td class="fs10"><?= $ticket->get("ouverture") ?></td>
        <td class="w200 tdPop fs10 mrlauto txt-center d-none flex
j-between a-center gap16"><div><?= $produit->get("design") ?><br><?=
$ticket->getHTML("demande") ?><br><a
href="afficher_ticket.php?idTicket=<?= $id ?>">Détail ⇛</a>
        <?php
            if($ticket->get("fermeture") != ""){
                ?>
                <br> cloturé le <?= $ticket->get("fermeture") ?> par <?
$ticket->get("par") ?>
                <?php
```

```
            }
        ?>
        </div><button class="tdClose btnPad">✖</button></td>
    </tr>
    <?php
}
?>
</tbody>
```

Histo_vente.php

```
<?php
// fragment : création de la liste des ventes
?>
<thead>
    <tr>
        <th class="fs12">Vendeur</th>
        <th class="fs12">Client</th>
        <th class="fs12">Produit</th>
        <th class="fs12">Date</th>
    </tr>
</thead>
<tbody>
<?php
foreach($liste as $id => $vente){
    $vendeur = $vente->getTarget("vendeur");
    $client = $vente->getTarget("client");
    $produit = $vente->getTarget("produit");
    ?>
    <tr class="histo">
        <td class="fs10"><?= $vendeur->getHTML("nom") ?></td>
        <td class="fs10"><?= $client->getHTML("nom") ?></td>
        <td class="fs10"><?= $produit->get("ref") ?></td>
        <td class="fs10"><?= $vente->get("date") ?></td>
        <td class="w200 tdPop fs10 mrlauto txt-center d-none flex
j-between a-center gap16"><div><?= $produit->get("design") ?><br> s/n: <?=
$vente->getHTML("sn") ?></div><button class="tdClose
btnPad">✖</button></td>
    </tr>
    <?php
}
```

```
?>
</tbody>
```

Liste_msg.php

```
<?php

// fragment constructeur de la liste des message d'un ticket

foreach($listeMsg as $id => $msg){
    $auteur = $msg->getTarget("auteur");
    ?>
        <div class="mt8">
            <p class="mt4"><?= $auteur->getHTML("prenom") ?> - <?=
$msg->get("date") ?></p>
            <p class="mt4"><?= $msg->getHTML("contenu") ?></p>
        </div>
    <?php
}
```

Liste_ticket.php

```
<?php
// $listeAttente : - (si espace tech) tout les tickets ouv/ec avec dernier
msg= client
//                 - (si espace client) tout les tickets ouv/ec avec
dernier msg different client
//$listeRepondu : - (si espace tech)  tout les tickets ouv/ec avec dernier
msg = client
//                - (si espace client) tout les tickets ouv/ec avec
dernier msg different client
?>


<h5 class="fs18">Tickets en attente de réponse </h5>

<ul class="mt16">
    <?php
    foreach($listeAttente as $id => $ticket){
        echo "<li class='mt8 fs14'>$id - ";
        if($espace === "technicien"){
```

```
            $client = $ticket->getTarget("client");
            $nomClient = $client->get("nom");
            echo "$nomClient - ";
        }
        $produit = $ticket->getTarget("produit");
        $refProduit = $produit->get("ref");
        $statut = $ticket->get("statut");
        $ouverture = $ticket->get("ouverture");
        echo "$refProduit - $statut - $ouverture - <a
href='afficher_ticket.php?idTicket=$id'>⇛</li>";
    }
    ?>
</ul>
<h5 class="mt40 fs18">Tickets ouvert ou en cours :</h5>
<ul class="mt16">
    <?php
    foreach($listeRepondu as $id => $ticket){
        echo "<li class='mt8 fs14'>$id - ";
        if($espace === "technicien"){
            $client = $ticket->getTarget("client");
            $nomClient = $client->get("nom");
            echo "$nomClient - ";
        }
        $produit = $ticket->getTarget("produit");
        $refProduit = $produit->get("ref");
        $statut = $ticket->get("statut");
        $ouverture = $ticket->get("ouverture");
        echo "$refProduit - $statut - $ouverture - <a
href='afficher_ticket.php?idTicket=$id'>⇛</a></li>";
    }
    ?>
</ul>
```

Select_produit_vente.php

```
<?php

// fragment : construit le select de la demande

?>
```

```
<option value=''>Sélection produit</option>
<?php
    foreach ($listproduit as $id => $produit) {
        ?>
        <option value='<?= $id ?>'><?= $produit->get("design") ?></option>
        <?php
    }
```

Select_produit.php

```
<?php

// fragment : construit le select de la demande

?>
<option value=''>Sélectionnez votre produit</option>
<?php
    foreach ($listAchat as $id => $vente) {
        $produit = $vente->getTarget("produit");
        $nomProduit = $produit->get("design");
        $idProd = $vente->get("produit");
        ?>
        <option value='<?= $idProd ?>'><?= $nomProduit ?></option>
        <?php
    }
```

Mails

Confirm_creation_client.php

```
<?php

// Template pour le mail de confirmation
// Paramètre : néant



?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta charset="UTF-8">
```

```
</head>
<body>
    <h2>Bonjour <?= $prenom ?> !</h2><br>
    <p>Bienvenu sur l'application ticketing, votre compte a été créé avec
succès !</p><br>
    <p>Vous pouvez maintenant vous connecter avec vos identifiants en
suivant ce lien : <a
href="http://tickets.alaugier.mywebecom.ovh/ticketing/afficher_accueil.php
">Connexion</a></p><br>
    <p>Vos identifiants :</p><br>
    <ul>
        <li>Email : <?= $mailTo ?></li>
        <li>Mot de passe : <?= $param["pwd"] ?></li>
    </ul><br>
    <p>Ce mot de passe est généré automatiquement, nous vous invitons à le
modifier via les paramètres de l'application.</p>
</body>
</html>
```

Pages

Accueil.php

```
<?php
// template : Affiche la page d'accueil de l'appli - si vendeur : btns
histo vente, modif mdp, modif espace, créer clt et enregistrer vente
//                                                  - si tech : btns modif
mdp, histo mes tickets, modif espace, liste des tickets ouv/ec avec
dernier message =client + liste des ticket ec avec dernier message tech de
plus de 72H (pour relance) (si id vendeur : surbrillance de ses clients)
//                                                  - si clt : btns modif
info, histo tickets, nv ticket, liste ticket en attente rep et tickets
ouvert
// parametre : $id_user : id de l'utilisateur connecté
//             $espace : espace défini à la connexion ou par l'utilisateur
//             $listeAttente : - (si espace tech) liste de tout les
tickets ouv/ec avec dernier message = client
//                             - (si espace client) liste de tout les
tickets ouv/ec avec dernier message different client
//             $listeRepondu : - (si espace tech) liste de tout les
tickets ouv/ec avec dernier message = client
```

```
//                              - (si espace client) liste de tout les
tickets ouv/ec avec dernier message different client
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body class="blue">
    <?php include "templates/fragments/header.php"; ?>
    <main class="w300 mrlauto">
        <?php
            if($modif == 1){
                echo "<p id='popModif' class='encart fs14 txt-center'>La
modification est bien prise en compte</p>";
            }
            if($idClt != 0){
                echo "<p id='popModif' class='encart fs14 txt-center'>La
création du client $prenomClt $nomClt est bien prise en compte</p>";
            }
            if($vente == 1){
                echo "<p id='popModif' class='encart fs14 txt-center'>La
vente est bien prise en compte</p>";
            }
            if($ticket == 1){
                echo "<p id='popModif' class='encart fs14
txt-center'>L'ouverture de votre ticket est bien prise en compte'</p>";
            }
            if($cloture == 1){
                echo "<p id='popModif' class='encart fs14 txt-center'>Le
ticket est clôturé'</p>";
            }
        ?>
        <!-- section commune au 3 espaces -->
        <section id="commun">
            <div class="flex j-between mt40">
```

```
                <div class="flex">
                    <p id="param">⚙️</p>
                    <h3><?= $user->get("prenom") ?></h3>
                </div>
                <?php
                    if($espace === "vendeur"){
                        echo "<a
href='afficher_historique.php?historique=vente'><button
class='btnPad'>Historique des ventes</button></a>";
                    }else if($espace === "technicien"){
                        echo "<a
href='afficher_historique.php?historique=ticketAll'><button
class='btnPad'>Tickets archivés</button></a>";
                    }else if($espace === "client"){
                        echo "<a
href='afficher_historique.php?historique=ticketPerso'><button
class='btnPad'>Mes tickets</button></a>";
                    }
                ?>
            </div>
            <?php if($espace === "client") echo "<div class='flex j-end
mt4'><a href='afficher_demande.php'>+ Nouveau ticket</a></div>"; ?>
        </section>
        <div id="menu" class="encart d-none">
            <a href="deconnecter.php"><button
class="btnPad">Deconnexion</button></a>
            <button id="closeMenu" class="btnPad">✖</button>
            <?php
                if($espace === "vendeur" || $espace === "technicien"){
                    echo "<a class='block mt16'
href='afficher_modif_info.php'><button class='btnPad'>Modifier mon mot de
passe</button></a>";
                }else if($espace === "client"){
                    echo "<a class='block mt16'
href='afficher_modif_info.php'><button class='btnPad'>Modifier mes
informations</button></a>";
                }
            ?>
        </div>
        <!-- section spécifique -->
```

```
        <section id="espace" class="mt40">
            <?php
                if($espace === "vendeur" || $espace === "technicien"){
                    include "templates/fragments/change_espace.php";
                }
                // appel le fragment correspondant à l'espace
                if($espace === "vendeur"){
                    include "templates/fragments/espace-vendeur.php";
                }else{
                    echo "<div id='tickets' class='mt40'></div>";
                }
            ?>
            </div>
        </section>
    </main>
    <script src="js/param.js" defer></script>
    <?php
        if($popup == 1){
            echo "<script src='js/modif.js' defer></script>";
        }
        if($espace === "vendeur"){
            echo "<script src='js/recherche.js' defer></script>";
        }else{
            echo "<script src='js/ticket.js' defer></script>";
        }
    ?>
</body>
</html>
```

Client.php

```
<?php
// template : affiche le formulaire de création d'un client avec champ
nom, prénom, mail mdp et btns vamlider/annuler
// parametre : $error : 1 si 1 champ n'est pas renseigné
//             $nomClient = 1 si vide ou $nom
//             $prenomClient = 1 si vide ou $prenom
//             $mailClient= 1 si vide ou $mail
//             $mdpClient= 1 si vide ou $mdp
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <main class="w300 mrlauto blue">
        <h2 class="fs20 mt40">Création client</h2>
        <form class="encart mt40" action="creer_client.php" method="POST">
            <div class="w200 mrlauto">
                <label class='w100p block fs16' for='nomClient'>Nom
:</label>
                <input class="w100p mt4" type='text' name='nomClient'
id='nomClient' value='<?= nl2br(htmlentities($nomClient)) ?>'>
                <?php
                    if($error == 1 && $nomClient == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un nom</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16' for='prenomClien'>Prenom
:</label>
                <input class="w100p mt4" type='text' name='prenomClient'
id='prenomClient' value='<?= nl2br(htmlentities($prenomClient)) ?>'>
                <?php
                    if($error == 1 && $prenomClient == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un prenom</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16' for='mailClient'>Mail
:</label>
```

```
                <input class="w100p mt4" type='email' name='mailClient'
id='mailClient' value='<?= nl2br(htmlentities($mailClient)) ?>'>
                <?php
                    if($error == 1 && $mailClient == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un mail</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16 flex j-between a-center">
                <input class="fs14" type="submit" value="Créer">
                <a class="fs12" href="afficher_accueil.php">Annuler</a>
            </div>
        </form>
    </main>
    <script src="js/client.js" defer></script>
</body>
</html>
```

Connexion.php

```
<?php
// template : Affiche la page de connexion : champ mail, mdp + btns oeil,
connexion et annuler + msg d'erreur si necessaire
// parametre : $error : 1 si erreur mdp ou login
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body class="blue">
    <?php include "templates/fragments/header.php"; ?>
    <main class="w300 mrlauto blue">
        <h2 class="fs20 mt40">Connexion</h2>
        <form class="encart mt40" action="connecter.php" method="POST">
            <div class="w200 mrlauto">
```

```
                <label class="w100p block fs16" for="login">Email</label>
                <input class="w100p mt4" type="text" name="login"
id="login">
            </div>
            <div class="w200 mrlauto mt16">
                <label class="block w100p fs16" for="mdp">Mot de
passe</label>
                <div class="w100p flex j-between mt4">
                    <input class="w150" type="password" name="mdp"
id="mdp">
                    <p id="btnMdp" class="w30 btn white txt-center">👁️</p>
                </div>
            </div>
            <div class="w200 mrlauto mt16 flex j-between a-center">
                <input class="fs14" type="submit" value="connexion">
                <a class="fs12" href="">Mot de passe oublié</a>
            </div>
            <?php
                if($error == 1){
                    ?>
                    <p class="w200 mt16 fs12 red mrlauto">Il y a une
erreur de saisie sur ton email ou ton mot de passe</p>
                    <?php
                }
            ?>
        </form>
    </main>
    <script src="js/connexion.js" defer></script>
</body>
</html>
```

Demande_assistance.php

```
<?php
// template : affiche le formulaire de création d'une demande d'assistance
avec produit concerné et message, + btns valider ou annuler
// parametre : $error : 1 si 1 champ n'est pas renseigné
//             $produit = 1 si pas renseigné ou le produit
//             $dde = 1 si pas renseigné ou le produit
//             $listeprod = liste des produit acheté par le client
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<?php include "templates/fragments/header.php"; ?>
    <main class="w300 mrlauto blue">
        <h2 class="fs20 mt40">Demande d'assistance</h2>
        <form class="encart mt40" action="envoyer_demande.php"
method="POST">
        <div class="w200 mrlauto">
                <label class='w100p block fs16' for='idProduit'>Sélection
produit :</label>
                <select class="w100p mt4" name='idProduit' id='idProduit'>
                    <?php include
"templates/fragments/select_produit.php"; ?>
                </select>
                <?php
                    if($error == 1 && $idProduit == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez sélectionner un produit</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16' for='dde'>Votre demande
:</label>
                <textarea class="w100p mt4" type='text' name='dde'
id='dde'><?= nl2br(htmlentities($demande)) ?></textarea>
                <?php
                    if($error == 1 && $demande == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez formuler votre demande</p>";
                    }
                ?>
```

```
            </div>
            <div class="w200 mrlauto mt16 flex j-between a-center">
                <input class="fs14" type="submit" value="Ouvrir le
ticket">
                <a class="fs12" href="afficher_accueil.php">Annuler</a>
            </div>
        </form>
    </main>
</body>
</html>
```

Detail_ticket.php

```
<?php
// template : affiche le détail d'un ticket : client, produit, ref,
demande, statut + btn fermer et repondre
//            ainsi qu'une zone d'affichage de la liste des message liés +
un msg de confirmation de fermeture du ticket via javascript (caché par
défaut)
// parametre : $ticket : détail d'un ticket
?>

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Document</title>
        <link rel="stylesheet" href="css/style.css">
    </head>
    <body class="blue">
        <?php include "templates/fragments/header.php"; ?>
        <main class="w300 mrlauto">
            <div class="flex j-between a-center mt40">
                <h2 class="fs20">Ticket <?= $idTicket ?></h2>
                <a href="afficher_accueil.php"><button
class="btnPad">Accueil</button></a>
            </div>
            <div class="flex j-between a-center mt40">
                <div>
```

```
                    <ul>
                        <?php
                            if($espace != "client") echo "<li>$nomClient
$prenomClient</li>";
                        ?>
                        <li class="mt8"><?= $designProd ?></li>
                        <li class="mt8"><?= $refProd ?></li>
                    </ul>
                </div>
                <div id="containConfirm">
                    <ul>
                        <li class="mt8"><?= $ticket->get("statut") ?></li>
                        <?php
                            if($ticket->get("statut") != "CLO") echo "<li
class='mt16'><button id='cloture' class='btnPad'>Cloturer</button></li>";
                        ?>
                    </ul>
                    <div id="confirm" class="encart d-none">
                        <h3 class="txt-center">Confirmer la clôture du
ticket</h3>
                        <div class="flex j-around mt40">
                            <button id="annule"
class="btnPad">Non</button>
                            <a href="fermer_ticket.php?idTicket=<?=
$idTicket ?>"><button class="btnPad">Oui</button></a>
                        </div>
                    </div>
                </div>
            </div>
            <p><?= $ticket->get("demande") ?></p>
            <form class="mt16" action="envoyer_message.php" method="POST">
                <input type="number" name="idTicket" id="idTicket"
class="d-none" value="<?= $idTicket ?>">
                <div>
                    <textarea class="w100p h60" name="contenu"
id="contenu"></textarea>
                </div>
                <div class="flex j-end">
                    <input class="mt8" type="submit" value="Répondre">
                </div>
```

```
            </form>
            <div id="listMsg" class="mt16">

            </div>
        </main>
        <script src="js/msg.js" defer></script>
</body>
</html>
```

Form_vente.php

```
<?php
// template : affiche le formulaire de création d'une vente avec champ
client, produit, sn, date et btns valider et annuler
// parametre : $produit : détail du produit vendu
//             $error : 1 si 1 champ n'est pas renseigné
//             $client= 1 si vide ou $client
//             $sn= 1 si vide ou $sn
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<?php include "templates/fragments/header.php"; ?>
    <main class="w300 mrlauto blue">
        <h2 class="fs20 mt40">Vente produit</h2>
        <form class="encart mt40" action="enregistrer_vente.php"
method="POST">
        <?php
            if($error == 1 && $clientInconnu == 1){
                echo "<p class='w200 mt4 mb16 fs12 red mrlauto'>Le client
est inconnu, veuillez créer un nouveau client - <a
href='afficher_client.php'><u>Créer un client</u></a></p>";
            }
        ?>
```

```
        <div class="w200 mrlauto">
                <label class='w100p block fs16' for='nomClientVente'>Nom
:</label>
                <input class="w100p mt4" type='text' name='nomClientVente'
id='nomClientVente' value='<?= nl2br(htmlentities($nomClientVente)) ?>'>
                <?php
                    if($error == 1 && $nomClientVente == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un Nom</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16'
for='prenomClientVente'>Prenom :</label>
                <input class="w100p mt4" type='text'
name='prenomClientVente' id='prenomClientVente' value='<?=
nl2br(htmlentities($prenomClientVente)) ?>'>
                <?php
                    if($error == 1 && $prenomClientVente == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un prénom</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16' for='mailClientVente'>Mail
:</label>
                <input class="w100p mt4" type='email'
name='mailClientVente' id='mailClientVente' value='<?=
nl2br(htmlentities($mailClientVente)) ?>'>
                <?php
                    if($error == 1 && $mailClientVente == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un mail</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16">
```

```
                <label class='w100p block fs16'
for='designProduitVente'>Produit :</label>
                <input class="w100p mt4" type='text'
name='designProduitVente' id='designProduitVente' value='<?=
nl2br(htmlentities($designProduitVente)) ?>'>
                <?php
                    if($error == 1 && $designProduitVente == ""){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez selectionner un produit</p>";
                    }
                ?>
            </div>
            <input class="d-none" type="number" name="idProduit"
id="idProduit" value="<?= $idProduit ?>">
            <div class="w200 mrlauto mt16">
                <label class='w100p block fs16' for='snVente'>N° de série
:</label>
                <input class="w100p mt4" type='text' name='snVente'
id='snVente' value='<?= nl2br(htmlentities($snVente)) ?>'>
                <?php
                    if($error == 1 && ($snVente == "" || $snIncorrect ==
1)){
                        echo "<p class='w200 mt4 fs12 red
mrlauto'>Veuillez saisir un n° de série valide</p>";
                    }
                ?>
            </div>
            <div class="w200 mrlauto mt16 flex j-between a-center">
                <input class="fs14" type="submit" value="Valider">
                <a class="fs12" href="afficher_accueil.php">Annuler</a>
            </div>
        </form>
    </main>
    <script src="js/recherche.js" defer></script>
</body>
</html>
```

Historique.php

```
<?php
// template : affiche l'historique : - des vente si espace vente
```

```
//                                  - des tickets traité si espace tech
//                                  - des tickets si espace client
// parametre : $histo : historique - des vente si espace vente
//                                  - des tickets traité si espace tech
//                                  - des tickets si espace client
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<?php include "templates/fragments/header.php" ?>
    <main class="w300 mrlauto blue">
        <div class="flex j-between a-center mt40">
            <h2 class="fs20">Historique <?= $histo ?></h2>
            <a class="fs12" href="afficher_accueil.php"><button
class="btnPad">Retour</button></a>
        </div>
        <div class="w100p encart mt40">
            <table class="mrlauto">
                <?php
                    if($espace == "vendeur"){
                        include "templates/fragments/histo_ventes.php";
                    }else if($espace == "technicien"){
                        include "templates/fragments/histo_tickets.php";
                    }else if($espace == "client"){
                        include
"templates/fragments/histo_tickets_client.php";
                    }
                ?>
            </table>
        </div>
    </main>
    <script src="js/historique.js" defer></script>
</body>
```

```
</html>
```

Modif_info.php

```
<?php
// template : affiche le formulaire de modification des infos : mdp
(ancien pour vérif bonne personne, nouveau et confirm + mail si user =
client)
// parametre : $id_user : id de l'utilisateur connecté
//             $mail : mail de l'utilisateur(si id est un client)
//             $error : si erreur de saisie
//                                 $mail(si  renseigné)
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php" ?>
    <main class="w300 mrlauto blue">
        <h2 class="fs20 mt40">Modification de mes information</h2>
        <form class="encart mt40" action="enregistrer_modif.php"
method="POST">
            <?php
            if($espace === "client"){
                ?>
                <div class="w200 mrlauto">
                    <label class='w100p block fs16' for='mail'>Mail
:</label>
                    <input class="w100p mt4" type='email' name='mail'
id='mail' value='<?= htmlentities($mail) ?>'>
                </div>
                <?php
            }
            ?>
            <div class="w200 mrlauto mt16">
```

```
                <label class="w100p block fs16" for="mdpold">Ancien mot de
passe :</label>
                <div class="w100p flex j-between mt4">
                    <input class="w150" type="password" name="mdpOld"
id="mdpOld">
                    <p id="btnMdpOld" class="w30 btn white
txt-center">👁️</p>
                </div>
            </div>
            <div class="w200 mrlauto mt16">
                <label class="w100p block fs16" for="mdpNew">Nouveau mot
de passe :</label>
                <div class="w100p flex j-between mt4">
                    <input class="w150" type="password" name="mdpNew"
id="mdpNew">
                    <p id="btnMdpNew" class="w30 btn white
txt-center">👁️</p>
                </div>
            </div>
            <div class="w200 mrlauto mt16">
                <label class="w100p block fs16"
for="mdpVerif">Vérification :</label>
                <div class="w100p flex j-between mt4">
                    <input class="w150" type="password" name="mdpVerif"
id="mdpVerif">
                    <p id="btnMdpVerif" class="w30 btn white
txt-center">👁️</p>
                </div>
            </div>
            <div class="w200 mrlauto mt16 flex j-between a-center">
                <input class="fs14" type="submit" value="Modifier">
                <a class="fs12" href="afficher_accueil.php">Annuler</a>
            </div>
        </form>
        <?php
            if($error == 1){
                ?>
                <p class="w200 mt16 fs12 red mrlauto">Il y a une erreur de
saisie sur ton ancien mot de passe ou la vérification de ton nouveau mot
de passe</p>
```

```
                <?php
            }
        ?>
    </main>
    <script src="js/modification.js" defer></script>
</body>
</html>
```

Controleurs

Afficher_accueil.php

```
<?php
// controleur : si pas d'utilisateur connecté : demande l'affichage de la
page de connexion (si error mdp ou login retourne $erroe = 1 sinon 0)
//              si utilisateur connecté, demande l'affichage de la page
d'accueil
//                                      - création des param du template
: $user : l'utilisateur connecté
//
$espace : espace défini à la connexion ou par l'utilisateur
// parametre : $_session[id] : id de l'utilisateur connecté
//             $_session[espace] : espace sur lequel ce situe
l'utilisateur
//             $_GET[error] : si il y a une erreur à l'identification
//             $_GET[client] : si un client a été créé
//             $_GET[vente] : si une vente a été effectué


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

// vérification si on est connecté ou non
include "utils/verif_connexion.php";

// récupération des parametre
$idUser = session_isconnected() ? session_idconnected() : 0;
$espace = isset($_SESSION["espace"]) ? $_SESSION["espace"] : "";
$modif = isset($_GET["modif"]) ? $_GET["modif"] : 0;
$idClt = isset($_GET["client"]) ? $_GET["client"] : 0;
$vente = isset($_GET["vente"]) ? $_GET["vente"] : 0;
$ticket = isset($_GET["ticket"]) ? $_GET["ticket"] : 0;
```

```php
$cloture = isset($_GET["cloture"]) ? $_GET["cloture"] : 0;
$noProd = isset($_GET["noProd"]) ? $_GET["noProd"] : 0;

// si un client a été créé, je rcupère l'objet utilisateur de ce client
if($idClt != 0){
    $client = new utilisateur($idClt);
    $nomClt = $client->getHTML("nom");
    $prenomClt = $client->getHTML("prenom");
}

if($modif == 1 || $vente == 1 || $idClt != 0 || $ticket == 1 || $cloture
== 1){
    $popup = 1;
}else{
    $popup = 0;
}

// traitement
$user = new utilisateur($idUser);
$produit = new produit();
$listproduit = $produit->listAll();


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

Afficher_client.php

```php
<?php
// controleur : demande l'affichage du formulaire de création d'un client
// parametre : $_GET[error] : 1 si 1 champ n'est pas renseigné
//             $_GET[nom] = "" si vide ou $nom
//             $_GET[prenom] = "" si vide ou $prenom
//             $_GET[mail]= "" si vide ou $mail


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

```php
// récupération des parametre
$error = isset($_GET["error"]) ? $_GET["error"] : 0;
$nomClient = isset($_GET["nomClient"]) ? $_GET["nomClient"] : "";
$prenomClient = isset($_GET["prenomClient"]) ? $_GET["prenomClient"] : "";
$mailClient = isset($_GET["mailClient"]) ? $_GET["mailClient"] : "";

// traitement
// pas de traitement

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

Afficher_demande.php

```php
<?php
// controleur : demande l'affichage du formulaire d'assistance
// parametre : error : si un champ est mal ou non rempli
//             idProduit - produit : le produit concerné par la dde
//             idProduit - le detail de la demande


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

// récupération des parametre
$error = isset($_GET["error"]) ? $_GET["error"] : 0;
$idProduit = isset($_GET["idProduit"]) ? $_GET["idProduit"] : "";
$demande = isset($_GET["dde"]) ? $_GET["dde"] : "";
$idUser = isset($_SESSION["id"]) ? $_SESSION["id"] : 0;

// traitement
$vente = new vente();
// construction des filtre
$filtre = ["client" => $idUser];
// requete et stockage
$listAchat = $vente->listAll($filtre); // a rajouter par la suite filtre
produit acheter il y a moins d'un an

if($idProduit > 0){
    $produitDde = new produit($idProduit);
```

```
    $nomProduitDde = $produitDde->get("design");
}
else{
    $nomProduitDde = "";
}

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

Afficher_form_vente.php

```
<?php
// controleur : demande l'affichahe du formulaire d'enregistrement d'une
vente
// parametre : $_GET[error] : 1 si 1 champ n'est pas renseigné
//             nomClientVente - prenomClientVente - mailClientVente (info
du client)
//             clientInconnu - 1 si recherche du client dans la base a
echoué
//             produitVente (id du produit vendu)
//             snVente (n°série du produit renseigné par le vendeur)
//             snIncorrect - 1 si incorrect


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


// récupération des parametre
if(isset($_POST["idProduit"])) $idProduit = $_POST["idProduit"];
else if(isset($_GET["idProduit"])) $idProduit = $_GET["idProduit"];
else $idProduit = 0;

if($idProduit == 0){
    header("Location: afficher_accueil.php?noProd=1");
}

$error = isset($_GET["error"]) ? $_GET["error"] : 0;
$nomClientVente = isset($_GET["nomClientVente"]) ? $_GET["nomClientVente"]
: "";
```

```php
$prenomClientVente = isset($_GET["prenomClientVente"]) ?
$_GET["prenomClientVente"] : "";
$mailClientVente = isset($_GET["mailClientVente"]) ?
$_GET["mailClientVente"] : "";
$clientInconnu = isset($_GET["clientInconnu"]) ? $_GET["clientInconnu"] :
0;
$snVente = isset($_GET["snVente"]) ? $_GET["snVente"] : "";
$snIncorrect = isset($_GET["snIncorrect"]) ? $_GET["snIncorrect"] : 0;


// traitement
$produit = new produit($idProduit);
$designProduitVente = $produit->get("design");



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

Afficher_historique.php

```php
<?php
// controleur : demande l'affichage du template historique - création des
param du template : $histo : historique - des vente si espace vente
//
- des tickets traité si espace tech
//
- des tickets si espace client
// parametre : $_session[id] : id de l'utilisateur connecté
//             $_session[espace] : espace sur lequel ce situe
l'utilisateur


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


// récupération des parametre
$idUser = session_isconnected() ? session_idconnected() : 0;
$espace = isset($_SESSION["espace"]) ? $_SESSION["espace"] : "";
```

```php
// traitement
// si espace vendeur - on récupère la liste de toute les vente
if($espace == "vendeur"){
    $vente = new vente();
    $liste = $vente->listAll([], ["+date"]);
    $histo = "des ventes";
}else if($espace == "technicien"){
    // sinon si espace technicien - on récupère la liste de tout les
tickets
    $ticket = new ticket();
    $liste = $ticket->listAll([], ["+ouverture"]);
    $histo = "des tickets";
}else if($espace == "client"){
    // sinon si espace client - on récupère la liste des ticket du client
    $ticket = new ticket();
    $liste = $ticket->listAll(["client" => $idUser], ["+ouverture"]);
    //$liste = $ticket->listePerso($idUser);
    $histo = "de mes tickets";
}

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

Afficher_modif_info.php

```php
<?php
// controleur : demande l'affichage du formulaire de modification des
infos - création des param du template : $id_user : id de l'utilisateur
connecté
//
$mail : mail de l'utilisateur(si id est un client)
//
$error
// parametre : id : id de l'utilisateur via session si utilisateur
connecté, ou get si premiere connexion
//             $_session[espace] : espace sur lequel ce situe
l'utilisateur


// initialisation
```

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


// récupération des parametre
$idUser = session_isconnected() ? session_idconnected() : 0;
$espace = isset($_SESSION["espace"]) ? $_SESSION["espace"] : "";

// traitement
$user = new utilisateur($idUser);
$mail = $user->get("mail");

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

Afficher_ticket.php

```php
<?php
// controleur : demande l'affichage d'un ticket : retourne $ticket :
détail d'un ticket + espace connecté (si client ou non) + pour
surveillance mqsg en ajax, on stock idTicket dans la session
// parametre : $_GET[idTicket] : id du ticket à afficher


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


// récupération des parametre
$espace = isset($_SESSION["espace"]) ? $_SESSION["espace"] : "";
$idTicket = isset($_GET["idTicket"]) ? $_GET["idTicket"] : 0;



// traitement
// instanciation du ticket concerné
$ticket = new ticket($idTicket);
// récupération des nom prénom client
$client = $ticket->getTarget("client");
$nomClient = $client->get("nom");
```

```php
$prenomClient = $client->get("prenom");
// récupération des design ref du ticket
$produit = $ticket->getTarget("produit");
$designProd = $produit->get("design");
$refProd = $produit->get("ref");
// sauvegarde idticket dans la session
$_SESSION["idTicket"] = $idTicket;

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

Changer_espace.php

```php
<?php
// controleur : change l'espace afficher via $_SESSION[espace] - vente si tech et tech si vente
// parametre : $_GET[espace] : espace affiché sur l'accueil


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


// récupération des parametre
$espace = isset($_GET["espace"]) ? $_GET["espace"] : "";

// traitement
$_SESSION["espace"] = $espace;

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