Projet : pizza
Cahier des charges
MyPizza : ma pizza sur mesure
Nous livrons des pizzas, et nous disposons déjà d'un site permettant de choisir des pizzas, des produits complémentaires
(desserts, boissons, …) et de gérer le passage et le paiement de la commande.
Nous lançons une offre de pizza sur mesure.
Les clients pourront choisir :
la taille (petite, grande, moyenne)
le type de pâte (épaisse ou fine)
la base (tomate ou crème)
les ingrédients
Bien sûr, nous souhaitons pouvoir gérer dans une base de données les choix à chaque étape (avec la description
détaillée, le prix, une photo….)
Nous souhaitons que nos clients puissent composer leur pizza sur mesure de manière très simple et très interactive, sur
une seule page, avec :
l'élaboration étape par étape
le récapitulatif de la pizza en cours d'élaboration, avec les photos et le prix
le détail de chaque choix lorsqu'on clique sur l'élément
la possibilité à tout moment de changer un choix (taille, type, base, ingrédients…)
Si l'internaute interrompt sa session et revient, nous souhaitons qu'il retrouve la pizza en cours d'élaboration.
Lorsque tout est choisi (avec au moins deux ingrédients), un bouton « commander » permet de finaliser la commande :
pour cela, vous devez enregistrer la pizza composée dans la base de données, et rediriger le client sur la page de
commande, c'est à dire sur l'url finaliser_commande?id=xx (l'id à donner est l'id de la pizza composée). Nous avons
une solution pour cette partie, vous ne devez donc pas développer la finalisation et le paiement de la commande.
Les options (tailles, types de pâtes, bases, ingrédients) seront stockées dans une base de données, et les photos
associées stockées dans des répertoires. Nous compléterons les tables correspondantes à la main.
Pour ces différents éléments, nous souhaitons gérer un nom, une description (facilement visible par les clients), et une
photo.

Css
Style.css

```
* {
```

```
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}

.back-white{background-color: #ffffffe0;}
.back-grey{background-color: #e0e0e0e8;}
.back-blue{background-color: #0288D1;}
.back-green{background-color: #4CAF50;}
.opt-select{background-color: #a5d6a7e0;}
.opt-unselect{background-color: #f0f0f0e1;}

.vert{color: #2E7D32;}
.white{color: #FFFFFF;}
.rouge{color: #D32F2F;}

.back-recap{
    background-color: #ffffff7c;
    backdrop-filter: blur(1px);
}

.mrlauto{
    margin-left: auto;
    margin-right: auto;
}
.mt20p{margin-top: 20%;}
.mt4{margin-top: 4px;}
.mt8{margin-top: 8px;}
.mt16{margin-top: 16px;}
.mt32{margin-top: 32px;}
.mt80{margin-top: 80px;}
.mr16{margin-right: 16px;}
.gap4{gap: 4px;}
.gap16{gap: 16px;}
.mb32{margin-bottom: 32px;}

.btnPad{padding: 4px 8px;}
.padRecap{padding: 2px 4px;}
.pad4{padding: 4px;}
.pad8{padding: 8px;}
```

```css
.pad16{padding: 16px;}

.wfit{width: fit-content;}
.wauto{width: auto;}
.w80p{width: 80%;}
.w100p{width: 100%;}
.w50{width: 50px;}
.w70{width: 70px;}
.w90{width: 90px;}
.w150{width: 150px;}
.w160{width: 160px;}
.w200{width: 200px;}
.w300{width: 300px;}

.fs10{font-size: 10px;}
.fs12{font-size: 12px;}
.fs14{font-size: 14px;}
.fs16{font-size: 16px;}
.fs18{font-size: 18px;}
.fs20{font-size: 20px;}
.fs24{font-size: 24px;}
.fs32{font-size: 32px;}

.h30{height: 30px;}
.h40{height: 40px;}
.h50{height: 50px;}
.h60{height: 60px;}
.h80{height: 80px;}
.h100{height: 100px;}
.h100v{height: 100vh;}
.h100v-80{height: calc(100vh - 80px);}

.flex{
    display: flex;
    flex-wrap: wrap;
}
.d-none{display: none;}
.block{display: block;}
.j-center{justify-content: center;}
.j-between{justify-content: space-between;}
```

```css
.j-around{justify-content: space-around;}
.j-end{justify-content: end;}
.a-center{align-items: center;}
.a-between{align-items: space-between;}
.txt-center{text-align: center;}

.b-none{border: none;}
.radius5{border-radius: 5px;}
.radius10{border-radius: 10px;}
.radius50p{border-radius: 50%;}

ul li {list-style: none;}
:visited, :active, :link{text-decoration: none;}

body{
    /*background-color: #FFFFFF;*/
    color: #212121;
    position: relative;
}

a {color: #2E7D32;}

.header{height: 80px;}

.oeil{
    width: 30px;
    height: 20px;
}

.oeil img{
    width: auto;
    height: 100%;
}

.recap{
    width: 90px;
    height: 45px;
    position: relative;
    border: solid 1px #E0E0E0;
    border-radius: 10px;
```

```
}

#recap-compo{height: 143px; align-items: space-between;}

.back{
    width: 100%;
    height: 100%;
    position: absolute;
    top: 0;
    z-index: -1;
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    opacity: 1;
    border-radius: 10px;
}
.petite{background-image: url("../assets/petite.jpg");}
.moyenne{background-image: url("../assets/moyenne.jpg");}
.grande{background-image: url("../assets/grande.jpg");}
.fine{background-image: url("../assets/fine.jpg");}
.epaisse{background-image: url("../assets/epaisse.jpg");}
.calzone{background-image: url("../assets/calzone.jpg");}
.tomate{background-image: url("../assets/tomate.jpg");}
.creme{background-image: url("../assets/creme.jpg");}
.mozzarella{background-image: url("../assets/mozzarella.jpg");}
.chevre{background-image: url("../assets/chevre.jpg");}
.cheddar{background-image: url("../assets/cheddar.jpg");}
.parmesan{background-image: url("../assets/parmesan.jpg");}
.camembert{background-image: url("../assets/camembert.jpg");}
.tomates_cerise{background-image: url("../assets/tomates_cerise.jpg");}
.comte{background-image: url("../assets/comte.jpg");}
.emmental{background-image: url("../assets/emmental.jpg");}
.fourme{background-image: url("../assets/fourme.jpg");}
.auvergne{background-image: url("../assets/auvergne.jpg");}
.rochebaron{background-image: url("../assets/rochebaron.jpg");}
.pepperoni{background-image: url("../assets/pepperoni.jpg");}
.jambon_cru{background-image: url("../assets/jambon_cru.jpg");}
.anchois{background-image: url("../assets/anchois.jpg");}
.knacky{background-image: url("../assets/knacky.jpg");}
.saumon{background-image: url("../assets/saumon.jpg");}
```

```css
.jambon{background-image: url("../assets/jambon.jpg");}
.chorizo{background-image: url("../assets/chorizo.jpg");}
.poulet{background-image: url("../assets/poulet.jpg");}
.hache{background-image: url("../assets/viande_hache.jpg");}
.bacon{background-image: url("../assets/bacon.jpg");}
.saucisse{background-image: url("../assets/saucisse.jpg");}
.champignon{background-image: url("../assets/champignon.jpg");}
.aubergine{background-image: url("../assets/aubergine.jpg");}
.courgette{background-image: url("../assets/courgette.jpg");}
.piment{background-image: url("../assets/piment.jpg");}
.miel{background-image: url("../assets/miel.jpg");}
.oeuf{background-image: url("../assets/oeuf.jpg");}
.poivrons{background-image: url("../assets/poivrons.jpg");}
.olives{background-image: url("../assets/olives.jpg");}
.tomates_seche{background-image: url("../assets/tomates_seche.jpg");}
.oignon{background-image: url("../assets/oignon.jpg");}
.basilic{background-image: url("../assets/basilic.jpg");}
#banniere{
    background-image: url("../assets/banniere_carte.jpg");
    background-size: cover;
    background-position: top;
    background-repeat: no-repeat;
}


.swiper {
    width: 300px;
    height: 340px;
    background-image: url("../assets/arriere_plan.jpg");
    background-size: cover;
    background-repeat: no-repeat;
}

#containIngredient{
    height: 275px;
    overflow-y: auto;
}

.blur{
    z-index: 2;
```

```
    position: absolute;
    top: 0;
    left: 0;
    background-color: #a5d6a734;
    backdrop-filter: blur(5px);
}


.arriere-plan{
    z-index: -1;
    position: absolute;
    top: 80px;
    left: 0;
    background-size: cover;
    background-repeat: no-repeat;
    opacity: 0.2;
}
#elaboration{
    background-image: url("../assets/arriere_elab.jpg");
}
#finalisation{
    background-image: url("../assets/arriere_finalisation.jpg");
}
#connexion{
    background-image: url("../assets/arriere_connexion.jpg");
}
#creation{
    background-image: url("../assets/arriere_plan_creation.jpg");
}

#affichage-mdp{
    background-color: rgba(255, 255, 255, 0.781);
    height: 20px;
}

#pop-oubli{
    backdrop-filter: blur(5px);
    position: absolute;
    z-index: 1;
    top: 80px;
```

```
    left: 0;
}
```

Javascript
connexion.js

```
// gestion de l'affichage ou masquage du mdp

// récupération des élément et zone d'affichage
let oeilConnect = document.getElementById("oeil-connect");
let mdp = document.getElementById("mdp");

oeilConnect.addEventListener("click", (e) => {
    if(mdp.type == "password"){
        mdp.type = "text";
        oeilConnect.innerHTML = `<img src="assets/oeil_ouvert_blanc.svg"
alt="image d'un oeil">`;
    }
    else{
        mdp.type = "password";
        oeilConnect.innerHTML = `<img src="assets/oeil_ferme_blanc.svg"
alt="image d'un oeil"></img>`;
    }
})


// gestion mdp oublié
// récupé bouton mdp-oubli
let btnMdpOubli = document.getElementById("mdp-oubli");
// récupération de la div à afficher
let popOubli = document.getElementById("pop-oubli");
// surveillance du bouton d'ouverture
btnMdpOubli.addEventListener("click", (e) => {
    // au clic, j'enleve d-none de la pop
    popOubli.classList.remove("d-none");
})
```

creation.js

```
// gestion de l'affichage ou masquage des mdp + controle des conditions
demandé pour les mot de passes
```

```
// récupération des élément et zone d'affichage
let oeilCrea = document.getElementById("oeil-crea");
let mdpCrea = document.getElementById("mdpCrea");
let oeilVerif = document.getElementById("oeil-verif");
let mdpVerif = document.getElementById("mdpVerif");
let mdpTot = document.getElementById("mdpTot")
let mdpMaj = document.getElementById("mdpMaj")
let mdpNbr = document.getElementById("mdpNbr")
let mdpSpe = document.getElementById("mdpSpe")

// gestion affichage/masquage
oeilCrea.addEventListener("click", (e) => {
    if(mdpCrea.type == "password"){
        mdpCrea.type = "text";
        oeilCrea.innerHTML = `<img src="assets/oeil_ouvert_blanc.svg"
alt="image d'un oeil">`;
    }
    else{
        mdpCrea.type = "password";
        oeilCrea.innerHTML = `<img src="assets/oeil_ferme_blanc.svg"
alt="image d'un oeil"></img>`;
    }
})
oeilVerif.addEventListener("click", (e) => {
    if(mdpVerif.type == "password"){
        mdpVerif.type = "text";
        oeilVerif.innerHTML = `<img src="assets/oeil_ouvert_blanc.svg"
alt="image d'un oeil">`;
    }
    else{
        mdpVerif.type = "password";
        oeilVerif.innerHTML = `<img src="assets/oeil_ferme_blanc.svg"
alt="image d'un oeil"></img>`;
    }
})

verifSaisie();
// surveillance du contenu du mdp
mdpCrea.addEventListener("input", (e) => {
     verifSaisie();
```

```
})


// fonction indépendante de surveillance du nombre total de caractère, et
de la présence d'un chiffre, d'une lettre maj et d'un caractère spécial
// rend l'instruction vert ou rouge selon qu'elle soit respecté ou non
function verifTotal(){
    // Minimum de 8 caractère
    if(mdpCrea.value.length >= 8){
        mdpTot.classList.remove("rouge");
        mdpTot.classList.add("vert");
    }else{
        mdpTot.classList.remove("vert");
        mdpTot.classList.add("rouge");
    }
}
function verifMajuscule(){
    // mini 1 lettre MAJ
    let regMaj = /[A-Z]/;
    if(regMaj.test(mdpCrea.value)){
        mdpMaj.classList.remove("rouge");
        mdpMaj.classList.add("vert");
    }else{
        mdpMaj.classList.remove("vert");
        mdpMaj.classList.add("rouge");
    }
}
function verifChiffre(){
    // mini 1 chiffre
    let regNbr = /\d/;
    if(regNbr.test(mdpCrea.value)){
        mdpNbr.classList.remove("rouge");
        mdpNbr.classList.add("vert");
    }else{
        mdpNbr.classList.remove("vert");
        mdpNbr.classList.add("rouge");
    }
}
function verifSpecial(){
    // mini 1 caractère special
```

```
    let regSpe = /[\W_]/;
    if(regSpe.test(mdpCrea.value)){
        mdpSpe.classList.remove("rouge");
        mdpSpe.classList.add("vert");
    }else{
        mdpSpe.classList.remove("vert");
        mdpSpe.classList.add("rouge");
    }
}

function verifSaisie(){
    // lance toute les vérification de saisie du mdp
    verifTotal();
    verifMajuscule();
    verifChiffre();
    verifSpecial();
}


// génération d'un mdp à la dde
let generateur = document.getElementById("generateur");
let affichageMdp = document.getElementById("affichage-mdp");

// surveille le click sur le bouton de generation
generateur.addEventListener("click", (e) => {
    // lance la génération du pwd puisutilise le retour pour afficher le
mdp dans la zone d'affichage
    genPwd();
})

function genPwd(){
    // fonction ajax : lance le controleur ajax de génération de mdp et
l'affiche
    // parametre : aucun
    // retour : un fichier json {"mdp" => mdp}

    fetch("generer_mdp.php")
    .then(resp =>{
        return resp.json();
    })
```

```
    .then(retour => {
        afficheMdp(retour.mdpGen);
    })
}

function afficheMdp(mdp){
    // role : affiche le mot de passe dans sa zone d'affichage
    // parametre : le mot de passe a afficher
    // retour : aucun

    affichageMdp.innerText = mdp;
}
```

Surveille_recap.js

```
// gestion fermeture des popup

// gestion de la fermeture de la div lessOpt
let lessOpt = document.getElementById("lessOpt");
let fermLessOpts = document.querySelectorAll(".fermLessOpt");
fermLessOpts.forEach(fermLessOpt => {
    fermLessOpt.addEventListener("click", (e) => {
        lessOpt.classList.add("d-none");
    })
})

// gestion de la fermeture de la div error
let error = document.getElementById("error");
let fermErrors = document.querySelectorAll(".fermError");
fermErrors.forEach(fermError => {
    fermError.addEventListener("click", (e) => {
        console.log("ferme");
        error.classList.add("d-none");
    })
})

// Gestion de l'affichage du recap des options de l apizza

// implémentation de la fonction
function afficheRecap(){
    // lance le controleur ajax d'affichage des options selectionné
```

```
    // parametre : aucun
    // retour, l'html à afficher

    fetch("afficher_recap.php")
    .then(resp =>{
        return resp.text();
    })
    .then(retour => {
        recap.innerHTML = retour;
    })
}

// récupération de la zone d'affichage du recap
let recap = document.getElementById("recap");

// lancement de la fonction au chargement de la page
afficheRecap();
setInterval(afficheRecap, 100);



// Gestion de l'ajout/suppression d'une option dans la base de donné

// implémentation de la fonction
function ajoutOption(option, categorie, idOption){
    // role : lance le controleur ajax d'ajout d'une option en lui
applicant les parametres
    // parametres : option - l'option concerné
    //              categorie - categorie de l'option selectionné
    //              idOption - id de l'option selectionné
    // retour : aucun

    option.classList.add("opt-select");

fetch(`surveiller_action.php?action=ajout&categorie=${categorie}&idOption=
${idOption}`)
}
function supOption(option, categorie, idOption){
    // role : lance le controleur ajax de suppression d'une option en lui
applicant les parametres
    // parametres : option - l'option concerné
```

```
    //              categorie - categorie de l'option selectionné
    //              idOption - id de l'option selectionné
    // retour : aucun

    option.classList.remove("opt-select");

fetch(`surveiller_action.php?action=suppression&categorie=${categorie}&idO
ption=${idOption}`)
}

function removeUnselects(){
    // role : enleve les classe opt-unselect sur les element qui l'ont
    // parametre : aucun
    // retour : aucun

    let unselects = document.querySelectorAll(".opt-unselect");
    unselects.forEach(unselect =>{
        unselect.classList.remove("opt-unselect");
    })
}

async function ingredientSelect(){
    // role : récupère le nombre d'ingrédient selectionné (nbre de
composition)
    // porametre : aucun
    // retour : un fichier json du type {nombre => valeur}

    return fetch("calculer_nombre_ingredient.php")
    .then(resp => {
        return resp.json();
    })
    .then(retour => {
        return retour;
    })
}

function recupIngredientMaxi(idOption){
    // role : retourne le nbre d'ingrédient maxi d'une pizza
    // parametre : idOption - id de l'option selectionné
    // retour : maxi - le nbre d'ingredient maxi selectionnable
```

```
    if(idOption == 1) return 5;
    else if(idOption == 2) return 7;
    else if(idOption == 3) return 9;
}

function recupTaille(idOption){
    // role : retourne le nbre d'ingrédient maxi d'une pizza
    // parametre : idOption - id de l'option selectionné
    // retour : maxi - le nbre d'ingredient maxi selectionnable

    if(idOption == 1) return "petite";
    else if(idOption == 2) return "moyenne";
    else if(idOption == 3) return "grande";
}

function afficheMsgRetrait(idOption, nbr){
    // role : affiche la div annoncant qu'il y a trop d'ingrédient pour la
pizza
    // parametre : aucun
    // retour : aucun

    // récupération de la zone d'affichage du message
    let msgRetrait = document.getElementById("msgRetrait");
    let msgTaille = document.getElementById("msgTaille");
    // retrait de la classe d-none de ladiv contenant l'affichage
    lessOpt.classList.remove("d-none");
    // modification du texte d'affichage
    let taille = recupTaille(idOption);
    msgTaille.innerText = `Attention, il y a trop d'ingrédient pour une
${taille} pizza`;
    msgRetrait.innerText = `Veuillez en retirer au moins ${nbr}`;
}

async function handleClick(option, categorie, idOption){
    // role : gère les actions suite à un click
    // parametre : option - l'option concerné
    //             categorie - la categorie de l'option
    //             idOption - l'id de l'option
    // retour : aucun
```

```
    // si ingredient
    if(categorie == "ingredient"){
        // si l'option est select
        if(option.classList.contains("opt-select")){
            // récupère le nombre d'ingrédient selectionné
            let ingredients = await ingredientSelect();
            // j'enleve la classe select et supOpption
            supOption(option, categorie, idOption);
            // si nbrSelect = 9
            if(ingredients.nombre == ingredients.maxi){
                // j'enleve classe select sur toutes les option qui ont la
class unselect
                removeUnselects();
            }else if(ingredients.nombre > ingredients.maxi){
                option.classList.add("opt-unselect");
            }
        }else if(!option.classList.contains("opt-select") &&
!option.classList.contains("opt-unselect")){
        // si l'option n'est pas select, et n'est pas unselect
            // récupère le nombre d'ingrédient selectionné
            let ingredients = await ingredientSelect();
            // si nbreSelect = 8
            if(ingredients.nombre == (ingredients.maxi-1)){
                // je grise tout les ingredient qui ne sont pas select
                options.forEach(option =>{
                    let categorie = option.dataset.categorie;
                    if(!option.classList.contains("opt-select") &&
categorie == "ingredient"){
                        option.classList.add("opt-unselect")
                    }
                })
                // j'enleve la classe unselect que l'on vient d'appliquer
                option.classList.remove("opt-unselect");
            }
            // j'ajoute l'option select ajoutOption
            ajoutOption(option, categorie, idOption);
        }
    }else{
    // sinon
```

```
        // si l'option est select
        if(option.classList.contains("opt-select")){
            // j'enleve la classe select et supOption
            supOption(option, categorie, idOption);
        }else{
        // si l'option n'est pas select
            // je supprime la classe select si une option est coché dans
la catégorie
            options.forEach(option => {
                if(option.dataset.categorie === categorie &&
option.classList.contains("opt-select")){
                    option.classList.remove("opt-select");
                }
            })
            // j'ajoute la classe select et ajoutOption
            ajoutOption(option, categorie, idOption);
            // si la categorie est taille
            if(categorie == "taille"){
                // on récupère le nombre d'ingrédient selectionné
                let ingredients = await ingredientSelect();
                // on récupère le nombre d'ingrédient maximum pour la
taille
                let maxi = recupIngredientMaxi(idOption);
                // si le nombre maxi est inférieur au nombre select
                if(ingredients.nombre < maxi){
                    // on efface les opt-unselect
                    options.forEach(option =>{
                        let categorie = option.dataset.categorie;
                        if(!option.classList.contains("opt-select") &&
categorie == "ingredient"){
                            option.classList.remove("opt-unselect")
                        }
                    })
                }else if(ingredients.nombre > maxi){
                // sinon, si nbr select > à nbre maxi
                    // on affiche un pop up pour dire qu'il faut
sélectionner x ingredient (la difference entre maxi et nbre select)
                    afficheMsgRetrait(idOption, ingredients.nombre -
maxi);
                }
```

```
            }
        }
    }
}

// récupération des élément à surveiller
let options = document.querySelectorAll(".option");

// lance l'action par option
options.forEach(option => {
    // on récupère data-catégorie
    let categorie = option.dataset.categorie;
    let idOption = option.dataset.ref;
    // je surveille le clic
    option.addEventListener("click", (e) =>{
        handleClick(option, categorie, idOption);
    });
})


// affichage du prix de la pizza en direct
// récupération de la zone d'affichage
let prixDirect = document.getElementById("prixDirect");
// fonction ajax de récupération du prix
function affichePrix(){
    // role : récupère le prix de la pizza via controleur ajac
    // parametre : aucun
    // retour : aucun
    fetch("calculer_prix.php")
    .then(resp => {
        console.log(resp)
        return resp.json();
    })
    .then(retour => {
        prixDirect.innerHTML = `<b>Prix : ${retour.prix} €</b>`;
    })
}
// affichage du prix au lancement de la page
affichePrix();
// lancement récurent de la pizza
```

```
setInterval(affichePrix, 100);
```

swiper.js

```
const swiper = new Swiper('.swiper', {
    // Optional parameters
    direction: 'horizontal',
    loop: true,

    // Navigation arrows
    // If we need pagination
    pagination: {
        el: '.swiper-pagination',
    },
  });
```

Swiper est une librairie que j'ai intégré au projet

Utils

Field

```
<?php

namespace Aldev\Utils;

// 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)
```

```php
    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

namespace Aldev\Utils;

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

```php
// 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";
include "Modeles/option.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/taille.php";
include "Modeles/type.php";
include "Modeles/base.php";
include "Modeles/pizza.php";
include "Modeles/ingredient.php";
include "Modeles/composition.php";
include "Modeles/utilisateur.php";
include "utils/session.php";


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

Model.php

```php
<?php

namespace Aldev\Utils;
```

```
/*

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)
```

```php
*/

class _model {


    // 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_pizza_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 $this->id();
                else if(isset($this->fields[$name])) 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(isset($this->fields[$name])) $this->set($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 = '\\Aldev\\Modeles\\';
                        $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();
                // 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(" AND ", $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 = '\\Aldev\\Modeles\\'; get_class retourne
déjà la class avec son namespace
                    $class = \get_class($this);
                    $objet = new $class();
                    $objet->loadFromTab($result);
                    $array[$objet->id()] = $objet;
                }

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

Session.php

```
<?php
```

```php
namespace Aldev\Utils;
use Aldev\Modeles\Utilisateur;

/*


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

    // parametrage de durée
    ini_set("session.gc_maxlifetime", 3600);
    ini_set('session.cookie_lifetime', 0);
    // 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....
        // ....
    }



    // 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();
    }


}

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

namespace Aldev\Utils;

/*

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($pwd) : crypte le mdp et le charge dans le champ "pwd" (nom à adapter au projet - mdp, pwd, etc..) de l'objet courant (condition : mini 8 caractere, dont 1 majuscule, 1 chiffre et 1 carctere spécial)
            sendMail($template, $param) : envoie un mail de confirmation avec le mot de passe auto $param = tableau indexé par les champs suivant : ["(detailMailTo)", "mailTo", "subject", "appli", "from", "reply"]
            loginVerify($login, $pwd) : verifie la concordance du mot de passe saisie et enregistré (nom du champ login à adapter au projet, idem pour le mdp)
            genToken() : genere un token unique en concaténant les information de l'utilisateur, avec son id et le time de la saisie, hashé, ajouter d'un chiffre aléatoire et transformé en valeur hexadecimal
            tokenVerify($token) : compare le token récupéré avec le token stocké dans la bdd, si la date de validité du token est toujours bonne et que le token correspond, alors on passe le compte au statut valide et on efface le token et sa date de validité

*/

class _user extends _model {

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

    function utilisateurExist($champ, $valeur){
        // vérifie dans la base de donnée si un utilisateur existe ou non avec le login renseigné
        // parametre : $login - l'information utilisateur utilisé pour ce connecté
        // retour : id de l'utilisateur ou 0 si pas d'utilisateur

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

```
        $param = [":valeur" => $valeur];

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

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

        // retour
        return $this->id();
    }

    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;
    }
```

```php
    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 sendMail($template, $param){
        // role : envoie un mail
        // 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
        $detail = isset($param["detailMailTo"]) ? $param["detailMailTo"] : "";
        $mailTo = "alaugier@mywebecom.ovh"; // ceci est pour les test, en condition réel, il faudra mettre $param["mailTo"]
        $to = "'$detail' <$mailTo>";
        // sujet
        $subject = $param["subject"];
        //expediteur
            $appli = $param["appli"];
            $from = $param["from"];
```

```php
            $reply = $param["reply"];
            // en tete
            $head = [];
            $head["From"] = "$appli <$from>";
            $head["reply-to"] = "$reply";
            // 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;
    }

    function genToken(){
        // role : genere un token unique en concaténant les information de l'utilisateur, avec son id et le time de la saisie, hashé, ajouter d'un chiffre aléatoire et transformé en valeur hexadecimal
        // parametre : aucun
        // retour : $token, le token généré

        // concaténation de la chaine de caractere
        $chaine = $this->nom . $this->prenom . $this->mail . time();
        // hashage
        $hash = hash("SHA256", $chaine);
        // ajout d'un chiffre aléatoire
        $hash .= random_bytes(16);
        // transformation en valeur hexadecimal
        $token = bin2hex($hash);

        // retour
        return $token;

    }

    function tokenVerify($token){
        // role : compare le token récupéré avec le token stocké dans la bdd, si la date de validité du token est toujours bonne et que le token correspond, alors on passe le compte au statut valide et on efface le token et sa date de validité
        // parametre : $token - le token à vérifier
        // retour : 0 ou id

        //construction
        $sql = "SELECT `id`, `token` FROM `$this->table` WHERE " . $this->makeFilter($token);
        $param = $this->makeParam($token);

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

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

        // retour
        if($this->is() && $this->token == $token) return $this->id();
        else return 0;
    }
}
```

Verif_connexion.php

```
<?php

namespace Aldev\Utils;

/*

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

Base.php

```
<?php

namespace Aldev\Modeles;

/*

gestion de la classe base
```

```
*/

class base extends option{

protected $table = "base";

}
```

Composition.php

```
<?php

namespace Aldev\Modeles;

/*

gestion de la classe composition

Utilisation :
    define() : définition des champs de la classe -> addField($name, $type
= NULL, $libelle = NULL, $link = NULL);
    suppression($id) : suppression de toute les composition dont le champ
pizza est $id
    compo($idPizzElab, $idOption) : ajoute une composition dans la base de
donnée
    supAll($id) : suppression de toute les composition dont le champ pizza
est $id
    sup($pizza, $option) : supprime la composition correspondant aux
parametre

*/

class composition extends \Aldev\Utils\_model{

    protected $table = "composition";

    protected function define(){
        // création des champs de la class
        $this->addField("pizza", "LINK", "Pizza", "pizza");
        $this->addField("ingredient", "LINK", "Inredient", "ingredient");
    }
```

```php
    function compo($idPizzElab, $idOption){
        // role : ajoute une composition dans la base de donnée
        // parametre : $idPizzElab - id de la pizza concerné
        //             $idOption - id de l'option a rattaché à la pizza
        // retour : aucun

        $this->pizza = $idPizzElab;
        $this->ingredient = $idOption;
        $this->insert();
    }

    // function suppression($id){ // inutile
    function supAll($id){
        // role : suppression de toute les composition dont le champ pizza
est $id
        // parametre : $id - id de la pizza en cours d'elaboration
        // retour : true false

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

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

        // récupération
        // aucun

        // retour
        return true;
    }

    function sup($pizza, $option){
        // role : supprime la composition correspondant aux parametre
        // parametre : idPizzElab - id de la pizza concerné
        //             idOption - id de l'option concerné
        // retour : true / false

        // construction
```

```
        $sql = "DELETE FROM `$this->table` WHERE `pizza` = :id AND
`ingredient` = :opt";
        $param = [":id" => $pizza, ":opt" => $option];

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

        // récupération
        // aucun

        // retour
        return true;
    }

    function compoExist($pizza, $ingredient){
        // role : récupère l'id de la composition recherché, si elle
existe
        // parametre : $pizza = id de la pizza en cours
        //             $ingredient = id de l'ingredient de la compo
        // retour : l'id de la compo ou 0

        // construction
        $sql = "SELECT `id` FROM `$this->table` WHERE `pizza` = :pizza AND
`ingredient` = :ingredient";
        $param = [":pizza" => $pizza, "ingredient" => $ingredient];
        // preparation/execution
        $req = $this->runSql($sql, $param);
        // récupération
        $id = $this->recoverReqSimple($req);
        // retour
        if($id != 0) return $id;
        else return 0;
    }
}
```

Ingredient.php

```
<?php

namespace Aldev\Modeles;
```

```php
/*

gestion de la classe ingredient

*/

class ingredient extends option{

    protected $table = "ingredient";

    protected function define(){
        parent::define();
        $this->addField("categorie");
    }
}
```

Option.php

```php
<?php

namespace Aldev\Modeles;

/*

Modeles de gestion généralisé de la classe option etendu de la classe
modèle (à étendre aux différentes classes)

Utilisation :
    define() : définition des champs de la classe -> addField($name, $type
= NULL, $libelle = NULL, $link = NULL);

*/

class option extends \Aldev\Utils\_model{

    protected function define(){
        // création des champs de la class
        // $this->addField("categorie"); inutile
        $this->addField("nom");
        $this->addField("description");
        $this->addField("prix");
```

```
        $this->addField("photo");
    }

}
```

Pizza.php

```
<?php

namespace Aldev\Modeles;

/*

gestion de la classe composition

Utilisation :
    define() : définition des champs de la classe -> addField($name, $type
= NULL, $libelle = NULL, $link = NULL);
    newPizz() : crée une pizza dans la bdd, la ratache à l'utilisateur et
lui fixe le statut EC
    pizzInit() : initialise la pizza en cours
    nettoyage() : supprime toute les pizza ec expirée de la base de donnée
    majTime() : maj le time de la pizza en cours
    majAction() : maj la date de la pizza

    Inutiles :
    idElaboration($id) : retourne l'id une pizza en cours d'elaboration,
(existante ou  nouvelle)
        pizzExist($id) : recherche dans la bdd si l'utilisateur à une
pizza en cours
        newPizz($id) : crée une pizza dans la bdd, la ratache à
l'utilisateur et lui fixe le statut EC

*/

class pizza extends \Aldev\Utils\_model{

    protected $table = "pizza";

    protected function define(){
        // création des champs de la class
```

```
        // $this->addField("utilisateur", $type = "LINK", $libelle =
"Utilisateur", $link = "utilisateur"); // inutiles
        $this->addField("utilisateur", "LINK", "Utilisateur",
"utilisateur");
        $this->addField("statut");
        $this->addField("taille", "LINK", "Taille", "taille");
        $this->addField("type", "LINK", "Type", "type");
        $this->addField("base", "LINK", "base", "base");
        $this->addField("time");


    }

/* inutile
    function idElaboration($id){
        // role : retourne l'id une pizza en cours d'elaboration,
(existante ou  nouvelle)
        // parametre : $id - id de l'utilisateur connecté
        // retour : $idPizza - l'id de la pizza en elaboration

        // je vérifie qu'une pizza en cours existe et retourne son id
        if($this->pizzExist() != 0) return $this->pizzExist();
        // sinon j'en crée une nouvelle et retourne  son id
        else return $this->newPizz($id);
    }

    function pizzExist($id){
        // role : recherche dans la bdd si l'utilisateur à une pizza en
cours
        // parametre : id de l'utilisateur connecté
        // retour : idPizz - id de la pizza en cours

        //construction
        $sql = "SELECT `id` FROM `$this->table` WHERE `utilisateur` = :id
AND `statut` = 'EC'";
        $param = [":id" => $id];
        // preparation/execution
        $req = $this->runSql($sql, $param);
        // récupération/ retour
        if ($this->recoverReqSimple($req)) return $this->id();
```

```
        else return 0;
    }
*/
    // function newPizz($id){ // inutile
    function newPizz(){
        // role : crée une pizza dans la bdd, la ratache à l'utilisateur
et lui fixe le statut EC
        // parametre : id de l'utilisateur connecté
        // retour : idPizz - id de la pizza en cours

        // chargement de l'utilisateur
        // $this->set("utilisateur", $id); // inutile
        // chargement du statut
        $this->statut = "EC";
        // chargement du time
        $this->majTime();
        // insertion dans la bdd
        $this->insert();
        $_SESSION["idPizzElab"] = $this->id();
        // retour de l'id
        return $this->id();
    }

    function pizzInit(){
        // role : initialise la pizza en cours
        // parametre : aucun
        // retour : aucun

        $this->taille = NULL;
        $this->type = NULL;
        $this->base = NULL;
        $this->majTime();
        $this->update();

        // instanciation d'un objet composition vierge
        $compo = new composition();
        // suppression des composition relatif à la pizza
        $compo->supAll($this->id);
    }
```

```
    function nettoyage(){
        // role : supprime toute les pizza ec expirée de la base de donnée
        // parametre : aucun
        // retour : aucun

        $timeActu = \time();
        $timeExpir = 3600;
        // récupère la liste de toutes les pizza expiré
        $liste = $this->listAll(["statut" => "EC"]);
        if(!empty($liste)){
            foreach($liste as $idPizza => $pizza){
                // pour chaque pizza concerné, si le time est inferieur a
1h, supprime leur composition
                if($pizza->time < ($timeActu - $timeExpir)){
                    $compo = new composition();
                    $compo->supAll($idPizza);
                    // supprime la pizza
                    $pizza->delete();
                }
            }
        }
    }

    function majTime(){
        // role : maj le time de la pizza en cours
        // parametre : id de la pizza en cours
        // retour : aucun

        // maj de la pizza avec le time
        $time = \time();
        $this->time = $time;
    }

    function majAction(){
        // role : maj la date de la pizza
        // parametre : aucun
        // retour : aucun

        $this->majTime();
        $this->update();
```

```
    }

    function infoPizz(){
        // role : renvoie les informations de la pizza
        // parametre : aucun
        // retour : les informations de la pizza dans un tableau indexé
par le nom du champ

        $taille = $this->getTarget("taille");
        $nomTaille = $taille->nom;
        $prixTaille = $taille->prix;
        $photoTaille = $taille->photo;
        $type = $this->getTarget("type");
        $nomType = $type->nom;
        $prixType = $type->prix;
        $photoType = $type->photo;
        $base = $this->getTarget("base");
        $nomBase = $base->nom;
        $prixBase = $base->prix;
        $photoBase = $base->photo;

        return ["nomTaille" => $nomTaille, "prixTaille" => $prixTaille,
"photoTaille" => $photoTaille, "nomType" => $nomType, "prixType" =>
$prixType, "photoType" => $photoType, "nomBase" => $nomBase, "prixBase" =>
$prixBase, "photoBase" => $photoBase];
    }
}
```

Taille.php

```
<?php

namespace Aldev\Modeles;

/*

gestion de la classe taille

*/

class taille extends option{
```

```
    protected $table = "taille";

    protected function define(){
        parent::define();
        $this->addField("mini");
        $this->addField("maxi");
    }
}
```

Type.php

```
<?php

namespace Aldev\Modeles;

/*

gestion de la classe type

*/

class type extends option{

protected $table = "type";

}
```

Utilisateur.php

```
<?php

namespace Aldev\Modeles;

/*

// ajout en plus du cahier des charges

gestion de la classe utilisateur
```

```
Utilisation :
    define() : définition des champs de la classe -> addField($name, $type
= NULL, $libelle = NULL, $link = NULL);

*/

class utilisateur extends \Aldev\Utils\_user{

    protected $table = "utilisateur";

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

    protected function define(){
        // création des champs de la class
        $this->addField("nom");
        $this->addField("prenom");
        $this->addField("mail");
        $this->addField("mdp");
        $this->addField("statut");
        $this->addField("token");
        $this->addField("validite");
    }
}
```

Templates
Fragments
Base.php

```
<?php

// fragment : affiche le choix de sélection de la base de la pizza

?>

<h3 class="wfit mrlauto mt4">La base</h3>
<div class="w100p mt4">
    <?php
        foreach($bases as $idBase => $base){
            ?>
```

```
                <div data-categorie="base" data-ref="<?= $idBase ?>"
class="option back-white mt8 radius10 w80p h80 mrlauto pad4 flex j-between
a-center <?php if($pizza->base == $idBase) echo "opt-select"; ?>">
                    <img class="h60 wauto radius50p" src="assets/<?=
$base->photo ?>.jpg" alt="photo d'une sauce <?= $base->photo ?>">
                    <p class="fs12 w160"><b><?= $base->nom ?> : <?=
$base->prix ?>€</b><br><?= $base->description ?></p>
                </div>
            <?php
        }
    ?>
</div>
```

Creation_mdp.php

```
<?php

// fragment : champ création mot de passe à intégrer dans un formulaire

?>

    <div class="w200 mrlauto mt16">
        <label class="block mrlauto wfit fs14" for="mdpCrea"><b>Mot de
passe</b></label>
        <div class="w200 mt4 flex j-between">
            <input class="w160" type="password" name="mdpCrea"
id="mdpCrea">
            <div id="oeil-crea" class="oeil flex j-center back-green
btnPad radius5 b-none"><img src="assets/oeil_ferme_blanc.svg" alt="image
d'un oeil"></div>
        </div>
    </div>
    <div id="instructionMdp" class="w200 mrlauto mt8">
        <p class="fs12">Le mot de passe doit contenir un minimum de :</p>
        <p id="mdpTot" class="fs12 mt4">- 8 caractères</p>
        <p id="mdpMaj" class="fs12">- 1 lettre majuscule</p>
        <p id="mdpNbr" class="fs12">- 1 chiffre</p>
        <p id="mdpSpe" class="fs12">- 1 caractère spécial</p>
        <p class="fs12 mt8">Tu peux aussi utiliser notre générateur de mot
de passe aléatoirement</p>
        <div class="w200 flex j-around a-center mt8">
```

```
            <p id="generateur" class="w50 white back-green btnPad radius5
b-none txt-center fs10">Générer</p>
            <p id="affichage-mdp" class="w70 btnPad radius5 b-none
txt-center fs10"></p>
        </div>
    </div>
    <div class="w200 mrlauto mt8">
        <label class="block mrlauto wfit fs12"
for="mdpVerif"><b>Vérification du mot de passe</b></label>
        <div class="mrlauto w200 mt4 flex j-between a-center">
            <input class="w160" type="password" name="mdpVerif"
id="mdpVerif">
            <div id="oeil-verif" class="oeil flex j-center back-green
btnPad radius5 b-none"><img src="assets/oeil_ferme_blanc.svg" alt="image
d'un oeil"></div>
        </div>
    </div>
    <?php if($error == 1 && $doubleMdp == 0) echo "<p class='w200 mt8 fs12
rouge mrlauto'>Veuillez saisir 2 mots de passe identiques</p>"; ?>
```

Form_connect.php

```
<?php

// ajout en plus du cahier des charges

// fragment : formulaire de connexion

?>

<form class="mt32" action="connecter.php" method="POST">
    <div class="w200 mrlauto">
        <label class="fs14 block mrlauto wfit"
for="mail"><b>Mail</b></label>
        <input class="mrlauto w200 mt8" type="text" name="mail" id="mail"
<?php if($idUtilisateur != 0) echo "value='$utilisateur->mail'"; ?>>
    </div>
    <div class="w200 mrlauto mt16">
        <label class="fs14 block mrlauto wfit" for="mdp"><b>Mot de
passe</b></label>
        <div class="w200 flex j-between mt8">
```

```
            <input class="w160" type="password" name="mdp" id="mdp">
            <div id="oeil-connect" class="oeil flex j-center back-green
btnPad radius5 b-none"><img src="assets/oeil_ferme_blanc.svg" alt="image
d'un oeil"></div>
        </div>
        <p id="mdp-oubli" class="fs10 vert txt-center mt8">Mot de passe
oublié ?</p>
    </div>
    <div class="flex j-center mt32">
        <input class="white back-green btnPad radius5 b-none txt-center"
type="submit" value="Connexion">
    </div>
    <?php if($error == 1) echo "<p class='w200 mt16 fs12 rouge mrlauto'>Il
y a une erreur de saisie sur ton mail ou mot de passe</p>"; ?>
</form>
```

Form_creation.php

```
<?php

// ajout en plus du cahier des charges

// fragment : affiche le formulaire de création d'un compte

?>

<form class="mt32" action="creer_compte.php" method="POST">
    <div class="w200 mrlauto flex j-between">
        <div class="w90">
            <label class="block mrlauto wfit fs14"
for="nom"><b>Nom</b></label>
            <input class="mrlauto w90 mt4" type="text" name="nom" id="nom"
value="<?= $nom ?>">
        </div>
        <div class="w90">
            <label class="block mrlauto wfit fs14"
for="prenom"><b>Prenom</b></label>
            <input class="mrlauto w90 mt4" type="text" name="prenom"
id="prenom" value="<?= $prenom ?>">
        </div>
    </div>
```

```
    <?php
        if($error == 1 && $nom == "" && $prenom == "") echo "<p
class='w200 mt4 fs12 rouge mrlauto'>Veuillez saisir un nom et un
prénom</p>";
        else if($error == 1 && $nom == "") echo "<p class='w200 mt4 fs12
rouge mrlauto'>Veuillez saisir un nom</p>";
        else if($error == 1 && $prenom == "") echo "<p class='w200 mt4
fs12 rouge mrlauto'>Veuillez saisir un prénom</p>";
    ?>
    <div class="w200 mrlauto mt8">
        <label class="block mrlauto wfit fs14"
for="mailCrea"><b>Mail</b></label>
        <input class="mrlauto w200 mt4" type="text" name="mailCrea"
id="mailCrea" value="<?= $mail ?>">
    </div>
    <?php if($error == 1 && $mail == "") echo "<p class='w200 mt4 fs12
rouge mrlauto'>Veuillez saisir un mail</p>"; ?>
    <?php include "templates/fragments/creation_mdp.php"; ?>
    <div class="flex j-center mt32">
        <input class="white back-green btnPad radius5 b-none txt-center"
type="submit" value="Créer">
    </div>
    <?php if($mailExist == 1) echo "<p class='w200 mt16 fs12 rouge
mrlauto'>Un compte est déjà créé avec ce mail</p>"; ?>
</form>
```

Form_modification.php

```
<?php

// fragment : formulaire de modification d'un mdp
// parametre : idUtilisateur

?>


<form class="mt32" action="modifier_mdp.php?idUtilisateur=<?=
$idUtilisateur ?>" method="POST">
    <?php include "templates/fragments/creation_mdp.php"; ?>
    <div class="flex j-center mt32">
```

```
        <input class="white back-green btnPad radius5 b-none txt-center"
type="submit" value="Modifier">
    </div>
</form>
```

Header.php

```
<?php

// fragment : header

?>

<div id="banniere" class="header w100p h80 back-grey flex a-center">
    <div class="flex a-center j-center w100p back-recap h80">
        <div class="w300 mrlauto flex a-center j-between">
            <h1 class="vert">Pizz'@ la carte</h1>
            <a href="deconnecter.php"><button class="white back-green
btnPad radius5 b-none">Quitter</button></a>
        </div>
    </div>
</div>
```

Ingredient.php

```
<?php

// fragment : affiche le choix de sélection de la taille de la pizza

?>

<h3 class="wfit mrlauto mt4">Les ingrédients</h3>
<div id="containIngredient" class="w100p mt4">
    <h4 class="txt-center mt8"><b>Fromages</b></h4>
    <?php
        $listeCompo = $compo->listAll(["pizza" => $idPizzElab]);
        foreach($ingredients as $idIngredient => $ingredient){
            if($ingredient->categorie == "fromage"){
                $compoSelect = $compo->compoExist($idPizzElab,
$idIngredient);
                include "templates/fragments/select_ingredient.php";
```

```
            }
        }
    ?>
    <h4 class="txt-center mt8"><b>Viandes</b></h4>
    <?php
        foreach($ingredients as $idIngredient => $ingredient){
            if($ingredient->categorie == "viande"){
                $compoSelect = $compo->compoExist($idPizzElab,
$idIngredient);
                include "templates/fragments/select_ingredient.php";
            }
        }
    ?>
    <h4 class="txt-center mt8"><b>Légumes</b></h4>
    <?php
        foreach($ingredients as $idIngredient => $ingredient){
            if($ingredient->categorie == "legume"){
                $compoSelect = $compo->compoExist($idPizzElab,
$idIngredient);
                include "templates/fragments/select_ingredient.php";
            }
        }
    ?>
    <h4 class="txt-center mt8"><b>Autres</b></h4>
    <?php
        foreach($ingredients as $idIngredient => $ingredient){
            if($ingredient->categorie == "autre"){
                $compoSelect = $compo->compoExist($idPizzElab,
$idIngredient);
                include "templates/fragments/select_ingredient.php";
            }
        }
    ?>
</div>
```

Mdp_oubli.php

```
<?php

// fragment : formulaire d'oubli mdp
```

```
?>


<div id="pop-oubli" class="w100p h100v-80 <?php if($mailOubli == 0) echo
"d-none "; ?>flex j-center a-center">
    <div>
        <h3 class="txt-center vert">Mot de passe oublié</h3>
        <?php
            if($mailOubli == 1 ){
                ?>
                    <p class='w300 mt4 fs12 rouge mrlauto'>Ce mail est
inconnu dans notre base de données, veuillez en saisir un autre pou créer
un compmte</p>
                <?php
            }
        ?>
        <form action="renvoyer_token.php" method="POST" class="mt32">
            <div>
                <label class="fs14 block txt-center"
for="mailOubli"><b>Mail</b></label>
                <input class="mrlauto w200 mt8" type="text"
name="mailOubli" id="mailOubli">
            </div>
            <div class="flex j-center mt16">
                <input class="white back-green btnPad radius5 b-none
txt-center" type="submit" value="Envoyer">
            </div>
        </form>
        <div class="w200 flex j-between mt32 a-center">
            <a href="afficher_elaboration.php"><button class="white
back-green btnPad radius5 b-none">Annuler</button></a>
            <a href="afficher_page_creation.php"><button class="white
back-green btnPad radius5 b-none">Créer un compte</button></a>
        </div>
    </div>
</div>
```

Recap.php

```
<?php
```

```
// ajout en plus du cahier des charges

// template : affiche le recap de la pizza -
taille-type-base-ingredient-prix
// parametre : aucun

?>


<div id="recap-pizza" class="flex j-between mt16">
    <div class="recap flex j-center a-center ">
        <div class="back <?= $photoTaille ?>"></div>
        <?php
            if(!is_null($infosPizz["nomTaille"])){
                ?>
                <p class='fs12 w100p vert back-recap padRecap
txt-center'><b><?= $infosPizz["nomTaille"] ?></b></p>
                <?php
            }
            else echo "<p>Taille</p>";
        ?>
    </div><div class="recap flex j-center a-center ">
    <div class="back <?= $photoType ?>"></div>
        <?php
            if(!is_null($infosPizz["nomType"])){
                ?>
                <p class='fs12 w100p vert back-recap padRecap
txt-center'><b><?= $infosPizz["photoType"] ?></b></p>
                <?php
            }
            else echo "<p>Type</p>";
        ?>
    </div><div class="recap flex j-center a-center ">
    <div class="back <?= $photoBase ?>"></div>
        <?php
            if(!is_null($infosPizz["nomBase"])){
                ?>
                <p class='fs12 w100p vert back-recap padRecap
txt-center'><b><?= $infosPizz["photoBase"] ?></b></p>
                <?php
```

```
            }
            else echo "<p>Base</p>";
        ?>
    </div>
</div>
<div id="recap-compo" class="flex j-between mt8">
    <?php
    foreach($listeCompo as $id => $compo){
        // récupération de 'lingrédient
        $ingredient = $compo->getTarget("ingredient");
        // récupération du nom
        $nom = $ingredient->get("nom");
        // récupération de txt photo
        $photo = $ingredient->get("photo");
        ?>
            <div class="recap flex j-center a-center ">
                <div class="back <?= $photo ?>"></div>
                <p class="vert fs12 back-recap padRecap w100p
txt-center"><b><?= $nom ?></b></p>
            </Div>
        <?php
    }

    for($i = 1; $i <= $divVide; $i++){
        ?>
            <Div class="recap flex j-center a-center opt-unselect">
                <p>/</p>
            </Div>
        <?php
    }

    ?>
</div>
```

Select_ingredient.php

```
<?php

// fragment : construit le bouton de sélection d'un ingredient
```

```
?>

<div data-categorie="ingredient" data-ref="<?= $idIngredient ?>"
class="option back-white mt8 radius10 w80p h40 mrlauto pad4 flex j-between
a-center <?php if($compoSelect != 0) echo "opt-select"; else
if(count($listeCompo) >= $taille->maxi) echo "opt-unselect" ?>">
    <img class="h30 wauto radius50p" src="assets/<?= $ingredient->photo
?>.jpg" alt="photo de <?= $ingredient->photo ?>">
    <p class="fs12"><b><?= $ingredient->nom ?></b></p>
    <p class="fs12"><b><?= $ingredient->prix ?>€</b></p>
</div>
```

Taille.php

```
<?php

// fragment : affiche le choix de sélection de la taille de la pizza

?>

<h3 class="wfit mrlauto mt4">La taille</h3>
<div class="w100p mt4">
    <?php
        foreach($tailles as $idTaille => $taille){
            ?>
                <div data-categorie="taille" data-ref="<?= $idTaille ?>"
class="option back-white mt8 radius10 w80p h80 mrlauto pad4 flex j-between
a-center <?php if($pizza->taille == $idTaille) echo "opt-select"; ?>">
                    <img class="h60 wauto radius50p" src="assets/<?=
$taille->photo ?>.jpg" alt="photo d'une <?= $taille->photo ?> pizza">
                    <p class="fs12 w160"><b><?= $taille->nom ?> : <?=
$taille->prix ?>€</b><br><?= $taille->description ?></p>
                </div>
            <?php
        }
    ?>
</div>
```

Type.php

```
<?php
```

```php
// fragment : affiche le choix de sélection du type de pâte de la pizza

?>

<h3 class="wfit mrlauto mt4">Le type de pâte</h3>
<div class="w100p mt4">
    <?php
        foreach($types as $idType => $type){
            ?>
                <div data-categorie="type" data-ref="<?= $idType ?>"
class="option back-white mt8 radius10 w80p h80 mrlauto pad4 flex j-between
a-center <?php if($pizza->type == $idType) echo "opt-select"; ?>">
                    <img class="h60 wauto radius50p" src="assets/<?=
$type->photo ?>.jpg" alt="photo d'une pizza <?= $type->photo ?>">
                    <p class="fs12 w160"><b><?= $type->nom ?> : <?=
$type->prix ?>€</b><br><?= $type->description ?></p>
                </div>
            <?php
        }
    ?>
</div>
```

Mails

Finalisation_mail.php

```php
<?php

// ajout en plus du cahier des charges

// template de mail : envoi un mail à l'utilisateur avec un recap de sa
pizza finalisé
//

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - récapitulatif pizza</title>
</head>
<body>
    <p>Bonjour <?= $param["prenom"] ?>,</p>
    <p>Merci d'avoir commandé chez pizz'@ la carte !</p>
    <p>Voici le récapitulatif de ta pizza personnalisée :</p>
    <p><b>- Taille de la pizza :</b> <?= $param["taille"] ?></p>
    <p><b>- Type de pâte :</b> <?= $param["type"] ?></p>
    <p><b>- Base :</b> <?= $param["base"] ?></p>
    <p><b>- Ingrédients :</b></p>
    <ul>
        <?php
        foreach($param["ingredients"] as $id => $nom){
            echo "<li>- $nom</li>";
        }
        ?>
    </ul>
    <p><b>- Prix total :</b> <?= $param["prix"] ?>€</p>
    <p>Merci, l'équipe Pizz'@ la carte</p>
</body>
</html>
```

Oubli_mail.php

```
<?php

// ajout en plus du cahier des charges

// template de mail : mail avec lien de vérification pointant vers le
formulaire de modification de mdp
//

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - Mot de passe oublié</title>
```

```
</head>
<body>
    <p>Bonjour <?= $param["prenom"] ?>,</p>
    <p>Pour créer un nouveau mot de passe, vérifie ton adresse e-mail en cliquant sur le lien ci-dessous :</p>
    <p><a href="http://pizza.alaugier.mywebecom.ovh/pizza/verifier_token.php?mdpOubli=1&token=<?= $param["token"] ?>">Pizz'@ la carte</a></p>
    <p>Si tu n'as pas demandé la modification de ton mot de passe, ignore cet e-mail.</p>
    <p>Merci,</p>
    <p>L'équipe Pizz'@ la carte</p>
</body>
</html>
```

Verification_mail.php

```
<?php

// ajout en plus du cahier des charges

// template de mail : vérification du mail utilisateur
//

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - vérification du mail</title>
</head>
<body>
    <p>Bonjour <?= $param["prenom"] ?>,</p>
    <p>Merci de t'être inscrit sur Pizz'@ la carte ! Pour compléter ton inscription et activer ton compte, vérifie ton adresse e-mail en cliquant sur le lien ci-dessous :</p>
    <p><a href="http://pizza.alaugier.mywebecom.ovh/pizza/verifier_token.php?token=<?= $param["token"] ?>">Pizz'@ la carte</a></p>
```

```
    <p>Si tu n'as pas créé de compte sur Pizz'@ la carte, ignore cet
e-mail.</p>
    <p>Merci,</p>
    <p>L'équipe Pizz'@ la carte</p>
</body>
</html>
```

Pages

Connexion.php

```
<?php

// ajout en plus du cahier des charges

// template : page de connexion de l'aplicaytion
// parametre : idPizzElab - id de la pizza en cours d'élaboration
// ajout : idUtilisateur - id de l'utilisateur qui a verifier son compte

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - connexion</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <div id="connexion" class="arriere-plan w100p h100v-80"></div>
    <?php include "templates/fragments/mdp_oubli.php"; ?>
    <main class="w300 mrlauto h100v-80 flex a-center j-center">
        <div>
            <?php
                if($creation == 1) echo "<p class='w200 vert fs12
txt-center mb32'>Compte créé avec succès !<br>Un e-mail de vérification
t'a été envoyé.</p>";
                if($verif == 1 ) echo "<p class='w200 vert fs12 txt-center
mb32'>Vérification réussie !<br>Ton compte est maintenant actif.</p>";
```

```
                if($echec == 1) echo "<a
href='renvoyer_verif.php?idUtilisateur=$idUtilisateur'><p class='w200 vert
fs12 txt-center mb32'>Le lien de vérification à expiré.<br>Clic <b>ici</b>
pour recevoir un nouveau lien de vérification</p></a>";
                if($renvoi == 1) echo "<p class='w200 vert fs12 txt-center
mb32'>Un e-mail de vérification t'a été envoyé.</p>";
                if($inactif == 1) echo "<a
href='renvoyer_verif.php?idUtilisateur=$idUtilisateur'><p class='w200 vert
fs12 txt-center mb32'>Ton compte est inactif.<br>Clic <b>ici</b> pour
recevoir un lien de vérification et le réactiver.</p></a>";
                if($majMdp == 1) echo "<p class='w200 vert fs12 txt-center
mb32'>La modification de ton mot de passe est bien prise en compte.</p>";
            ?>
            <h2 class="txt-center vert">Connexion</h3>
            <?php include "templates/fragments/form_connect.php"; ?>
            <h4 class="txt-center mt80">Pas encore de compte ?</h3>
            <div class="flex j-center mt32"><a
href="afficher_page_creation.php"><button class="white back-green btnPad
radius5 b-none">Créer un compte</button></a></div>
        </div>
    </main>
    <script src="js/connexion.js" defer></script>
</body>
</html>
```

Elaboration.php

```
<?php

// template : Affiche la page d'élaboration d'une pizza :
//              - 1 visuel de chaque option avec image en fond
//              - Prix
//              - btn réinitialiser et finaliser cde
//              - zone de choix de l'option et bouton option suivant ou
précédent

?>

<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - Elaboration de la pizza</title>
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" href="librairies/swiper/swiper-bundle.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <div id="elaboration" class="arriere-plan w100p h100v-80"></div>
    <main>
        <section id="recap" class="w300 mrlauto"></section>
        <section class="w300 mrlauto">
            <div class="w100p flex j-between mt16 a-center">
                <a href="reinitialiser.php"><button id="raz" class="white
back-green btnPad radius5 b-none">Réinitialiser</button></a>
                <p id="prixDirect"><b>Prix : - €</b></p>
                <a href="valider.php"><button id="valide" class="white
back-green btnPad radius5 b-none">Valider</button></a>
            </div>
        </section>
        <section id="select-option" class="w300 mrlauto">
            <!-- Slider main container -->
            <div class="swiper radius10 mt16">
            <!-- Additional required wrapper -->
                <div class="swiper-wrapper">
                    <!-- Slides -->
                    <div class="swiper-slide"> <?php include
"templates/fragments/taille.php"; ?> </div>
                    <div class="swiper-slide"> <?php include
"templates/fragments/type.php"; ?> </div>
                    <div class="swiper-slide"> <?php include
"templates/fragments/base.php"; ?> </div>
                    <div class="swiper-slide" id="ingredient"> <?php
include "templates/fragments/ingredient.php"; ?> </div>
                </div>
                <!-- If we need pagination -->
                <div class="swiper-pagination"></div>
            </div>
        </section>
    </main>
```

```
    <?php
    if($error == 1){
        ?>
        <div id="error" class="blur flex a-center w100p h100v">
            <div class="w300 back-grey mrlauto radius10 pad16">
                <div class="flex j-end"><button class="fermError white
back-green btnPad radius5 b-none">×</button></div>
                <ul class="mt32 back-white pad8 radius5">
                    <li><b>Avant de valider, sélectionnez :</b></li>
                    <li class="mt8">- La taille de votre pizza,</li>
                    <li class="mt4">- Le type de pâte souhaté,</li>
                    <li class="mt4">- Sa base,</li>
                    <li class="mt4">- Un minimum de <?= $taille->mini ?>
ingrédients</li>
                    <li class="mt4">- Et un maximum de <?= $taille->maxi
?> ingrédients</li>
                </ul>
                <div class="flex j-center mt32"><button class="fermError
white back-green btnPad radius5 b-none">Modifier ma pizza</button></div>
            </div>
        </div>
        <?php
    }
    ?>
    <div id="lessOpt" class="blur flex a-center w100p h100v d-none">
        <div class="w300 back-grey mrlauto radius10 pad16">
            <div class="flex j-end"><button class="fermLessOpt white
back-green btnPad radius5 b-none">×</button></div>
            <div class="mt32 back-white pad8 radius5">
                <p id="msgTaille">Attention, il y a trop d'ingrédient pour
cette taille</p>
                <p id="msgRetrait" class="mt8">Veuillez en retirer</p>
            </div>
            <div class="flex j-center mt32"><button class="fermLessOpt
white back-green btnPad radius5 b-none">Modifier ma pizza</button></div>
        </div>
    </div>
    <script src="js/surveille_recap.js" defer></script>
    <script src="librairies/swiper/swiper-bundle.js" defer></script>
    <script src="js/swiper_init.js" defer></script>
```

```
</body>
</html>
```

Finalisation.php

```
<?php

// ajout en plus du cahier des charges

// template : Affiche la page d'élaboration d'une pizza :
//              - 1 visuel de chaque option avec image en fond
//              - Prix
//              - btn réinitialiser et finaliser cde
//              - zone de choix de l'option et bouton option suivant ou
précédent

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - récapitulatif de la pizza finalisée</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <div id="finalisation" class="arriere-plan w100p h100v-80"></div>
    <main class="w300 mrlauto">
        <h2 class="mt32">Pizza en préparation</h2>
        <h3 class="mt32">Pizza n°<?= $pizza->id() ?> : <?= $prixTotal ?>
€</h3>
        <div class="flex a-center mt16">
            <p class="mr16"><b>Taille :</b> <?= $infosPizz["nomTaille"]
?></p>
            <img class="h30 wauto radius50p" src="assets/<?=
$infosPizz["photoTaille"] ?>.jpg" alt="photo d'une <?=
$infosPizz["photoTaille"] ?> pizza">
        </div>
        <div class="flex a-center mt4">
```

```
            <p class="mr16"><b>Type de pâte :</b> <?=
$infosPizz["nomType"] ?></p>
            <img class="h30 wauto radius50p" src="assets/<?=
$infosPizz["photoType"] ?>.jpg" alt="photo d'une pâte <?=
$infosPizz["photoType"] ?>">
        </div>
        <div class="flex a-center mt4">
            <p class="mr16"><b>Base :</b> <?= $infosPizz["nomBase"] ?></p>
            <img class="h30 wauto radius50p" src="assets/<?=
$infosPizz["photoBase"] ?>.jpg" alt="photo d'une sauce <?=
$infosPizz["photoBase"] ?>">
        </div>
        <p class="mt4"><b>Ingrédients :</b></p>
        <?php
        foreach($listeCompo as $id => $compo){
            // récupération de 'lingrédient
            $ingredient = $compo->getTarget("ingredient");
            ?>
            <div class="flex a-center mt4">
                <img class="h30 wauto radius50p mr16" src="assets/<?=
$ingredient->photo ?>.jpg" alt="photo d'une sauce <?= $ingredient->photo
?>">
                <p><?= $ingredient->nom ?></p>
            </div>
            <?php
        }
        ?>
        <a href="afficher_elaboration.php"><button class="white back-green
btnPad radius5 b-none mt32">Céer une nouvelle pizza</button></a>
    </main>
    </body>
</html>
```

Modification_mdp.php

```
<?php

// ajout en plus du cahier des charges

// template : affiche le formulaire de modification d'un mdp
// parametre : idUtilisateur
```

```
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - création d'un compte</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <div id="creation" class="arriere-plan w100p h100v-80"></div>
    <main class="w300 mrlauto h100v-80 flex a-center j-center">
        <div>
            <h2 class="txt-center vert">Modification mot de passe</h3>
            <?php include "templates/fragments/form_modification.php"; ?>
            <div class="flex j-center mt32"><a
href="afficher_elaboration.php"><button class="white back-green btnPad
radius5 b-none">Annuler</button></a></div>
        </div>
    </main>
    <script src="js/creation.js" defer></script>
</body>
</html>
```

Page_creation.php

```
<?php

// ajout en plus du cahier des charges

// template : affiche le formulaire de création d'un compte
// parametre : aucun

?>

<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pizz'@ la carte - création d'un compte</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <?php include "templates/fragments/header.php"; ?>
    <div id="creation" class="arriere-plan w100p h100v-80"></div>
    <main class="w300 mrlauto h100v-80 flex a-center j-center">
        <div>
            <h2 class="txt-center vert">Création d'un compte</h3>
            <?php include "templates/fragments/form_creation.php"; ?>
            <div class="flex j-center mt32"><a
href="afficher_elaboration.php"><button class="white back-green btnPad
radius5 b-none">Annuler</button></a></div>
        </div>
    </main>
    <script src="js/creation.js" defer></script>
</body>
</html>
```

Controleurs

Afficher_elaboration.php

```
<?php

// controleur : Affiche la page d'élabporation d'une pizza, crée ou charge
la pizza en cours + supprime les pizza expiré
// parameres : error
//             idPizzElab
// Parametre ajouté : creation - information permettant de savoir si un
compte a été créé ou non
//                    verif - info si 'lon vien de la page de verification
ou non
//                    idUtilisateur - en lien avec la verif, id de
l'utilisateur qui est passé par la verification de son mail

// initialisation

// use function Aldev\Utils\session_idconnected; // inutile
```

```php
// use function Aldev\Utils\session_isconnected; // inutile

use Aldev\Modeles\composition;

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


use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
$creation = isset($_GET["creation"]) ? $_GET["creation"] : 0;
$verif = isset($_GET["verif"]) ? $_GET["verif"] : 0;
$echec = isset($_GET["echec"]) ? $_GET["echec"] : 0;
$idUtilisateur = isset($_GET["idUtilisateur"]) ? $_GET["idUtilisateur"] :
0;
$renvoi = isset($_GET["renvoi"]) ? $_GET["renvoi"] : 0;
if($idUtilisateur != 0) $utilisateur = new
Aldev\Modeles\utilisateur($idUtilisateur);
$inactif = isset($_GET["inactif"]) ? $_GET["inactif"] : 0;
$mailOubli = isset($_GET["mailOubli"]) ? $_GET["mailOubli"] : 0;
$majMdp = isset($_GET["majMdp"]) ? $_GET["majMdp"] : 0;
// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile

// récupération
$error = isset($_GET["error"]) ? $_GET["error"] : 0;
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;

// traitement
// si une pizza en cours d'élaboration existe je récupère l'id, sinon j'en
crée une nouvelle
if($idPizzElab != 0){
    $pizza = new \Aldev\Modeles\pizza($idPizzElab);
    if($pizza->statut == "EC"){
        $pizza->majAction();
    } else{
        $pizza = new \Aldev\Modeles\pizza();
```

```
        $pizza->newpizz();
    }
}
else {
    $pizza = new \Aldev\Modeles\pizza();
    $pizza->newpizz();
}
$pizza->nettoyage();

// construit les liste des options
$compo = new composition();

$ingredient = new \Aldev\Modeles\ingredient();
$ingredients = $ingredient->listAll();
$type = new \Aldev\Modeles\type();
$types = $type->listAll();
$base = new \Aldev\Modeles\base();
$bases = $base->listAll();
// récupération de l'objet taille pour le calcul du nombre d'ingredient
disponible
$taille = new \Aldev\Modeles\taille($pizza->taille);
$tailles = $taille->listAll();


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

Afficher_modification_mdp.php

```
<?php

// controleur : demande l'affichage du formulaire de modification du mot
de passe
// parametre : idUtilisateur - id de l'uutilisateur qui change son mdp

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

// récupération
$idUtilisateur = isset($_GET["idUtilisateur"]) ? $_GET["idUtilisateur"] :
0;
```

```php
$error = isset($_POST["error"]) ? $_POST["error"] : 0;

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

Afficher_page_creation.php

```php
<?php

// ajout en plus du cahier des charges

// controleur : dde l'affichage du formulaire de création d'un compte
// parametre : error doublemdp- nom prenom mail renseigné par
l'utilisateur

// initialisation


include "utils/init.php";// vérification d'un utilisateur connecté, sinon
redirection vers la connexion

// récupération
$error = isset($_GET["error"]) ? $_GET["error"] : 0;
$nom = isset($_GET["nom"]) ? $_GET["nom"] : "";
$prenom = isset($_GET["prenom"]) ? $_GET["prenom"] : "";
$mail = isset($_GET["mail"]) ? $_GET["mail"] : "";
$doubleMdp = isset($_GET["doubleMdp"]) ? $_GET["doubleMdp"] : 1;
$mailExist = isset($_GET["mailExist"]) ? $_GET["mailExist"] : 0;

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

Afficher_recap.php

```php
<?php

// controleur ajax : surveille et affiche les option de la pîzza
(taille-type-base-ingredient-prix)
// parametre : idPizzElab - id de la pizza en cours

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

```php
// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile

// récupération
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;

// traitement
// récupération nom et prix des options de la pizza
$pizza = new \Aldev\Modeles\pizza($idPizzElab);
$pizza->majAction();
$infosPizz = $pizza->infoPizz();
$photoTaille = !is_null($infosPizz["photoTaille"]) ?
$infosPizz["photoTaille"] : "opt-unselect";
$photoType = !is_null($infosPizz["photoType"]) ? $infosPizz["photoType"] :
"opt-unselect";
$photoBase = !is_null($infosPizz["photoBase"]) ? $infosPizz["photoBase"] :
"opt-unselect";

// récupération de la liste des ingredient
$composition = new \Aldev\Modeles\composition();
$listeCompo = $composition->listAll(["pizza" => $idPizzElab]);
//calcul prix total ingrédient
$prixIngredient = 0;
foreach($listeCompo as $id => $compo){
    // récupération de 'lingrédient
    $ingredient = $compo->getTarget("ingredient");
    // récupération du nom
    $prix = $ingredient->prix;
    $prixIngredient += $prix;
}
// calcul du nombre de div vide à afficher
$divVide = 9 - count($listeCompo);
```

```php
$prixTotal = $infosPizz["prixTaille"] + $infosPizz["prixType"] +
$infosPizz["prixBase"] + $prixIngredient;

// affichage
include "templates/fragments/recap.php";
```

Calculer_nombre_ingredient.php

```php
<?php

// controleur ajax : retourne le nombre de composition de la pizza et son
nombre maxi
// parametre : $idPizzElab : id de la pizza en cours

// initialisation

use Aldev\Modeles\composition;
use Aldev\Modeles\pizza;
use Aldev\Modeles\taille;

include "utils/init.php";

use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile



// récupération
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;

// traitement
// nombre compo
$compo = new composition();
$liste = $compo->listAll(["pizza" => $idPizzElab]);
$nbr = count($liste);
// nombre maxi
$pizza = new pizza($idPizzElab);
$taillePizza = $pizza->taille;
$taille = new taille($taillePizza);
```

```php
$maxi = $taille->maxi;

// retour
echo json_encode(["nombre" => $nbr, "maxi" => $maxi]);
```

Calculer_prix.php

```php
<?php

// controleur ajax : calcule le prix de la pizza en cours
// parametre : $idPizzElab - id de la pizza en corus

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

// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile

// récupération
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;

// traitement
$pizza = new \Aldev\Modeles\pizza($idPizzElab);
$pizza->majAction();
$infosPizz = $pizza->infoPizz();
// récupération de la liste des ingredient
$composition = new \Aldev\Modeles\composition();
$listeCompo = $composition->listAll(["pizza" => $idPizzElab]);
//calcul prix total ingrédient
$prixIngredient = 0;
foreach($listeCompo as $id => $compo){
    // récupération de 'lingrédient
    $ingredient = $compo->getTarget("ingredient");
    // récupération du nom
    $prix = $ingredient->prix;
    $prixIngredient += $prix;
```

```
}
$prixTotal = $infosPizz["prixTaille"] + $infosPizz["prixType"] +
$infosPizz["prixBase"] + $prixIngredient;

// retour
echo json_encode(["prix" => $prixTotal]);
```

Connecter.php

```
<?php

// ajout en plus du cahier des charges

// controleur : compare le mail et mdp envoyé et les compare avec la base
de donnée pour connecter l'utilisateur (à développer pour la bonne
réalisation des test)
//              Crée une nouvelle pizza au statut en cours d'elaboration
ou charge l'existante si il y en a une
// paramètres : mail : mail de l'utilisateur
//              mdp : mot de passe de l'utilisateur

// initialisation

use function Aldev\Utils\session_connect;

include "utils/init.php";



// récupération
$mail = isset($_POST["mail"]) ? $_POST["mail"] : "";
$mdp = isset($_POST["mdp"]) ? $_POST["mdp"] : "";

// traitement
// instanciation d'un objet utilisateur vierge
$utilisateur = new Aldev\Modeles\utilisateur();
// vérification de la correspondance des identifiant (récupération de
l'id)
$idUtilisateur = $utilisateur->loginVerify($mail, $mdp);

// si l'id est égal à 0, echec authentification je renvoi sur la page
connexion
```

```php
if($idUtilisateur == 0){
    header("Location: afficher_elaboration.php?error=1");
    exit;
}

$utilisateur = new \Aldev\Modeles\utilisateur($idUtilisateur);
if($utilisateur->statut == 0){
    header("Location:
afficher_elaboration.php?inactif=1&idUtilisateur=$idUtilisateur");
    exit;
}
// sinon, je connecte l'utilisateur
session_connect($idUtilisateur);

// affichage
// (redirection vers la page d'elaboration)
header("Location: afficher_elaboration.php");
```

Creer_compte.php

```php
<?php

// ajout en plus du cahier des charges

// controleur : crée l'utilisateur dans la base de donnée - génére un
token, fixe sa durée de validité et env un mail de vérification
// parametre : nom - prenom - mail - mdp  de l'utilisateur

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

// récupération
$nom = isset($_POST["nom"]) ? $_POST["nom"] : "";
$prenom = isset($_POST["prenom"]) ? $_POST["prenom"] : "";
$mail = isset($_POST["mailCrea"]) ? $_POST["mailCrea"] : "";
$mdpCrea = isset($_POST["mdpCrea"]) ? $_POST["mdpCrea"] : "";
$mdpVerif = isset($_POST["mdpVerif"]) ? $_POST["mdpVerif"] : "";
if($mdpCrea == "" || $mdpVerif == "" || $mdpCrea != $mdpVerif) $doubleMdp
= 0;
else $doubleMdp = 1;
```

```php
// vérification si un utilisateur existe ou non avec ce mail
$utilisateur = new \Aldev\Modeles\utilisateur();
$utilisateurExist = $utilisateur->utilisateurExist("mail", $mail);
if($utilisateurExist != 0){
    header("Location: afficher_page_creation.php?mailExist=1");
    exit;
}

// traitement
if($nom == "" || $prenom == "" || $mail == "" || $doubleMdp == 0){
    header("Location:
afficher_page_creation.php?error=1&nom=$nom&prenom=$prenom&mail=$mail&doub
leMdp=$doubleMdp");
    exit;
}

// chargement d'un nouvel utilisateur avec ses informations
$utilisateur->nom = $nom;
$utilisateur->prenom = $prenom;
$utilisateur->mail = $mail;
$utilisateur->setPwd($mdpCrea);
// chargement du statut
$utilisateur->statut = 0;
// génération et chargement d'un token
$token = $utilisateur->genToken();
$utilisateur->token = $token;
// récupération et chargement d'un time
$time = \time();
$utilisateur->validite = $time;
// insertion
$utilisateur->insert();
// envoi du mail
$utilisateur->sendMail("templates/mails/verification_mail.php", ["prenom"
=> $prenom, "token" => $token, "detailMailTo" => "$prenom $nom", "mailTo"
=> "$mail", "subject" => "Finalise ton inscription à Pizz'@ la carte",
"appli" => "Pizz'à la carte", "from" => "alaugier@mywebecom.ovh", "reply"
=> "alaugier@mywebecom.ovh"]);

// affichage
header ("Location: afficher_elaboration.php?creation=1");
```

Deconnecter.php

```php
<?php

// controleur : deconnecte l'utilisateur (à développer pour la bonne
réalisation des test)
// parametre : aucun

// intitialisation

use function Aldev\Utils\session_deconnect;

include "utils/init.php";

// récupération
// aucun

// traitement
session_deconnect();

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

Fincaliser_commande.php

```php
<?php

// controleur de test - finalisartion commande
// parametres : id pizza

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


// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
```

```php
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile


// récupération
$id = isset($_GET["id"]) ? $_GET["id"] : 0;

// traitement
$pizza = new \Aldev\Modeles\pizza($id);
$infosPizz = $pizza->infoPizz();
// récupération de la liste des ingredient
$composition = new \Aldev\Modeles\composition();
$listeCompo = $composition->listAll(["pizza" => $id]);

//calcul prix total ingrédient (et préparation du tableau de nom
d'ingredient à envoyer par mail)
$prixIngredient = 0;
$nomsIngredients = [];
foreach($listeCompo as $id => $compo){
    // récupération de 'lingrédient
    $ingredient = $compo->getTarget("ingredient");
    // récupération du nom
    $prix = $ingredient->prix;
    $prixIngredient += $prix;
    $nomsIngredients[] = $ingredient->nom;
}

$prixTotal = $infosPizz["prixTaille"] + $infosPizz["prixType"] +
$infosPizz["prixBase"] + $prixIngredient;

// préparation et envoi d'un mail récapitulatif
$param = ["prenom" => $utilisateurConnecte->prenom, "taille" =>
$infosPizz["nomTaille"], "type" => $infosPizz["nomType"], "base" =>
$infosPizz["nomBase"], "ingredients" => $nomsIngredients, "prix" =>
$prixTotal, "detailMailTo" => "$utilisateurConnecte->prenom
$utilisateurConnecte->nom", "mailTo" => "$utilisateurConnecte->mail",
"subject" => "Récapitulatif de ta pizza personnallisée", "appli" =>
"Pizz'à la carte", "from" => "alaugier@mywebecom.ovh", "reply" =>
"alaugier@mywebecom.ovh"];
$utilisateurConnecte->sendMail("templates/mails/finalisation_mail.php",
$param);
```

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

Generer_mdp.php

```php
<?php

// controleur ajax : génere un mdp aléatoire
// parametre : aucun

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

// traitement
$utilisateur = new \Aldev\Modeles\utilisateur();
$mdpGen = $utilisateur->genPwd();

echo json_encode(["mdpGen" => $mdpGen]);
```

Modifier_mdp.php

```php
<?php

// controleur : modifie un mdp dans la bdd
// parametre : idUtilisateur
//             mdp

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

// récupération
$idUtilisateur = isset($_GET["idUtilisateur"]) ? $_GET["idUtilisateur"] :
0;
$mdpCrea = isset($_POST["mdpCrea"]) ? $_POST["mdpCrea"] : "";
$mdpVerif = isset($_POST["mdpVerif"]) ? $_POST["mdpVerif"] : "";
if($mdpCrea == "" || $mdpVerif == "" || $mdpCrea != $mdpVerif) $doubleMdp
= 0;
else $doubleMdp = 1;

// traitement
```

```php
// traitement
if($doubleMdp == 0){
    header("Location:
afficher_modification_mdp.php?error=1&doubleMdp=$doubleMdp&idUtilisateur=$
idUtilisateur");
    exit;
}
$utilisateur = new Aldev\Modeles\utilisateur($idUtilisateur);
$utilisateur->setPwd($mdpCrea);
$utilisateur->statut = 1;
$utilisateur->update();

// affichage
header("Location: afficher_elaboration.php?majMdp=1");
```

Reinitialiser.php

```php
<?php

// controleur : supprime les options de la pizza à modifier
// parametres : aucun

// initialisation

// use function Aldev\Utils\session_idconnected; //inutile
// use function Aldev\Utils\session_isconnected; // inutile

include "utils/init.php";

// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile

// récupération
// $idUser = session_isconnected() ? session_idconnected() : 0; // inutile
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;
```

```php
// traitement
// Chargement de la pizza
$pizza = new \Aldev\Modeles\pizza($idPizzElab);
// remise à 0 des options
$pizza->pizzInit();

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

Renvoyer_token.php

```php
<?php

// controleur : renvoi un nouveau token, si la vérification a echoué (a cause d'un échec)
// parametre : idUtilisateur

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

// récupération
$mail = isset($_POST["mailOubli"]) ? $_POST["mailOubli"] : "";

// traitement
$utilisateur = new \Aldev\Modeles\utilisateur();

$idUtilisateur = $utilisateur->utilisateurExist("mail", $mail);

if($idUtilisateur == 0){
    header("Location: afficher_elaboration.php?mailOubli=1");
    exit;
}

// maj statut, token, validite et update
$utilisateur->statut = 0;
$token = $utilisateur->genToken();
$time = \time();
$utilisateur->token = $token;
$utilisateur->validite = $time;
$utilisateur->update();
```

```php
// construction parametre et envoi du mail
$param = ["prenom" => $utilisateur->prenom, "token" => $token,
"detailMailTo" => "$utilisateur->prenom $utilisateur->nom", "mailTo" =>
$utilisateur->prenom, "subject" => "Crée un nouveau mot de passe pour
Pizz'@ la carte", "appli" => "Pizz'@ la carte", "from" =>
"alaugier@mywebecom.ovh", "reply" => "alaugier@mywebecom.ovh"];
$utilisateur->sendMail("templates/mails/oubli_mail.php", $param);

// affichage
header("Location:
afficher_elaboration.php?renvoi=1&idUtilisateur=$idUtilisateur");
```

Surveiller_action.php

```php
<?php

// controleur ajax : ajoute l'option selectionné à la pizza
// parametre : $idOption : id de l'option à ajouter

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

// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile

// récupération
$idOption = isset($_GET["idOption"]) ? $_GET["idOption"] : 0;
$categorie = isset($_GET["categorie"]) ? $_GET["categorie"] : "";
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;
$action = isset($_GET["action"]) ? $_GET["action"] : "";

// traitement
// instanciation d'un objet pizza chargé par l'id de la pizz en cours
$pizza = new Aldev\Modeles\pizza($idPizzElab);
if($categorie == "ingredient"){
    // instanciation d'un nouvel objet composition vierge
```

```
    $composition = new Aldev\Modeles\composition();
    // appel de la méthode de création ou suppression d'une composition
    if($action == "ajout") $composition->compo($idPizzElab, $idOption);
    else if($action == "suppression") $composition->sup($idPizzElab,
$idOption);
}else{
    // je charge l'id de l'option dans le champs concerné
    if($action == "ajout") $pizza->$categorie = $idOption;
    else if($action == "suppression") $pizza->$categorie = NULL;
}

// je met à jour la pizza avec le time de l'action
$pizza->majAction();

// retour
// aucun
```

Valider.php

```
<?php

// controleur ajax : vérifie que la pizza est éligible et redirige vers
l'url de finalisation
// parametres : aucun

// initialisation

// use function Aldev\Utils\session_idconnected; //inutile
// use function Aldev\Utils\session_isconnected; // inutile

use Aldev\Modeles\composition;
use Aldev\Modeles\taille;

include "utils/init.php";

// vérification d'un utilisateur connecté, sinon redirection vers la
connexion
use function Aldev\Utils\session_idconnected;
use function Aldev\Utils\session_isconnected;
include "utils/verif_connexion.php"; // inutile
$idUser = session_isconnected() ? session_idconnected() : 0; // inutile
```

```php
// récupération
// $idUser = session_isconnected() ? session_idconnected() : 0; // inutile
$idPizzElab = isset($_SESSION["idPizzElab"]) ? $_SESSION["idPizzElab"] :
0;

// vérifie qu'il y a une base, une taille et un type sélectionné ainsi que
3 ingrédients minimum, puis redirige si ok sinon demande d'ajouter ce
qu'il manque
// charge la pizza
$pizza = new Aldev\Modeles\pizza($idPizzElab);
// récupère la liste des commposition
$compo = new composition();
$listeCompo = $compo->listAll(["pizza" => $idPizzElab]);
$taille = new taille($pizza->taille);

// redirection
if(!is_null($pizza->taille) && !is_null($pizza->type) &&
!is_null($pizza->base) && count($listeCompo) >= $taille->mini &&
count($listeCompo) <= $taille->maxi){
    $pizza->statut = "FIN";
    $pizza->update();
    header("Location: finaliser_commande.php?id=$idPizzElab");
}
else header("Location: afficher_elaboration.php?error=1");
```

Verifier_token.php

```php
<?php

// ajout en plus du cahier des charges

// controleur : vérifie l'existance du token et sa validité, si ok, valide
le compte, et affiche la page de connexion
// parametre : token de verification

// initialisation

use Aldev\Modeles\utilisateur;
```

```php
include "utils/init.php";

// récupération
$token = isset($_GET["token"]) ? $_GET["token"] : 0;
$mdpOubli = isset($_GET["mdpOubli"]) ? $_GET["mdpOubli"] : 0;



// traitement
// on instancie un nouvel utilisateur
$utilisateur = new utilisateur();
// on récupère l'utilisateur qui possede le token
$utilisateur->utilisateurExist("token", $token);
$idUtilisateur = $utilisateur->id();

if($idUtilisateur == 0){
    header("Location: afficher_elaboration.php?noToken=1");
    exit;
}
if($mdpOubli == 1){
    // sinon, on récupère le time de validité
    $time = \time();
    if($utilisateur->validite < $time){
        // si le time actu est inferieur au time de validité, on affiche
la page connexion avec msg compte vérifié
        header("Location:
afficher_modification_mdp.php?idUtilisateur=$idUtilisateur");
    }else{
        // sinon on redirige vers page de création avec msg la
verification a expiré
        header("Location: afficher_elaboration.php?echec=1");
    }
}else{
    // si le statut est valide (1) on redirige vers la page de connexion
    if($utilisateur->statut == 1){
        header("Location:
afficher_elaboration.php?idUtilisateur=$idUtilisateur");
    }else{
        // sinon, on récupère le time de validité
        $time = \time();
        if($utilisateur->validite < $time){
```

```
            // si le time actu est inferieur au time de validité, on passe
le statut a 1 (valide) et on affiche la page connexion avec msg compte
vérifié
            $utilisateur->statut = 1;
            $utilisateur->update();
            header("Location:
afficher_elaboration.php?verif=1&idUtilisateur=$idUtilisateur");
        }else{
            // sinon on redirige vers page de création avec msg la
verification a expiré
            header("Location: afficher_elaboration.php?echec=1");
        }
    }
}
```

Toute la partie connexion de l'utilisateur, gestion mdp est un ajout de ma part afin de rendre l'application plus complete