

# DOCUMENTATION TECHNIQUE

(2TAK HARDWARE Symphony Version)

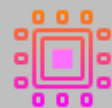


16 MAI



NOM DE LA SOCIÉTÉ

Créé par : MASVIDAL ADRIEN



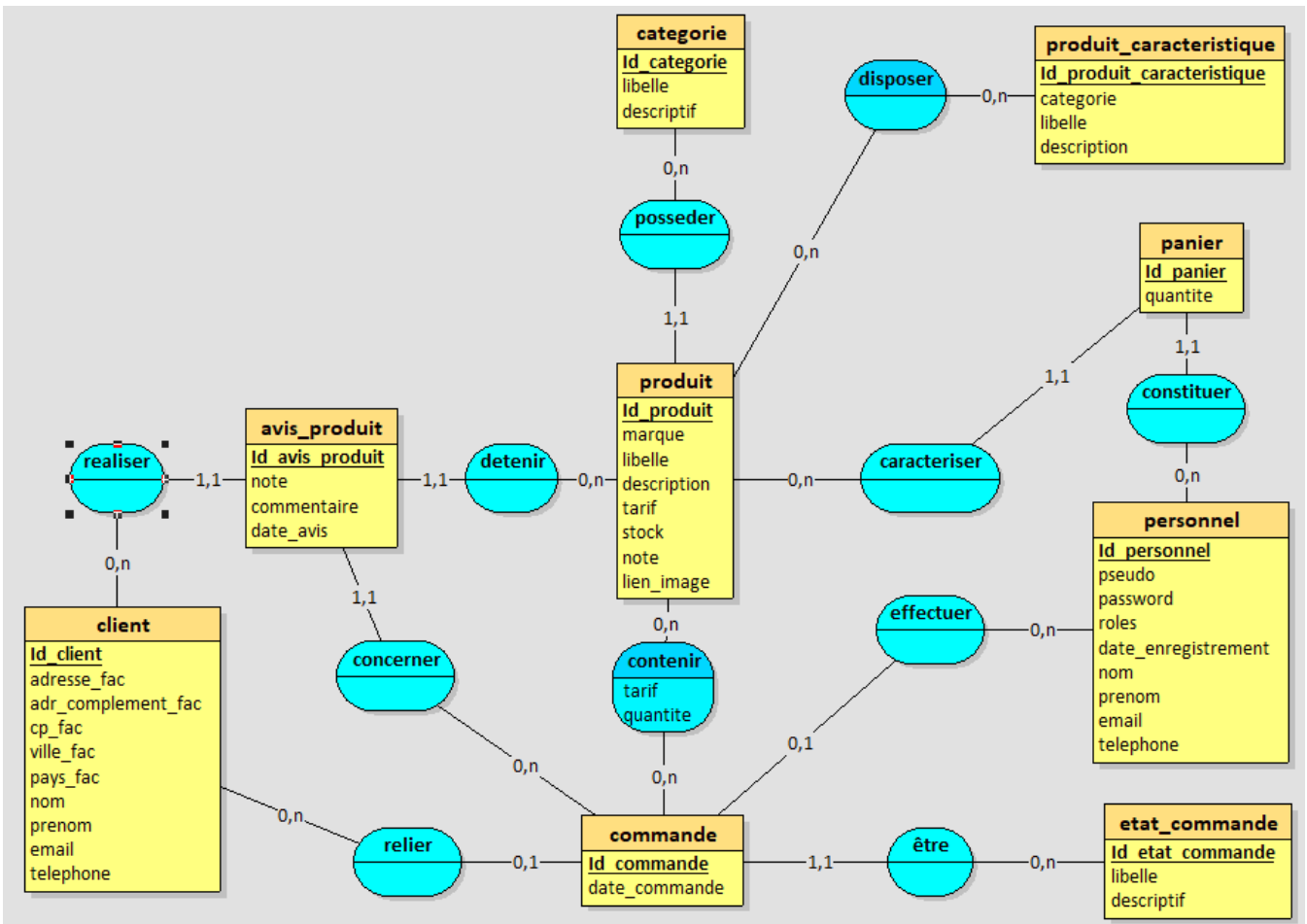
2TAK  
HARDWARE



## Table des matières

Base de données.....	3
MCD :.....	3
Schéma SQL existant : .....	4
Outils logiciels .....	5
Général .....	5
Gestion des styles (CSS, JavaScript) .....	5
API .....	6
Back Office.....	7
Services de paiement .....	8
Style .....	9
Card Produit .....	9
Graphiques .....	9

## MCD :





# Outils logiciels

## Général

2TAK HARDWARE Symfony version est un site web e-commerce proposant des services API pour l'application Android de gestion du contenu.

Ce site web est réalisé avec la version 5.1.8 de Symfony.

## Gestion des styles (CSS, JavaScript)

Les fichiers CSS et JavaScript du projet sont gérés avec la librairie JavaScript *'Webpack Encore'* proposé par la [documentation](#) Symfony.

```
var Encore = require('@symfony/webpack-encore');

// Manually configure the runtime environment if not already configured set by the "encore" command.
// It's useful when you use tools that rely on webpack.config.js file.
if (!Encore.isRuntimeEnvironmentConfigured()) {
    Encore.configureRuntimeEnvironment(process.env.NODE_ENV || 'dev');
}

Encore
    // Directory where compiled assets will be stored
    .setOutputPath('public/build/')
    // public path used by the web server to access the output path
    .setPublicPath('/build')
    // only needed for CDN's or sub-directory deploy
    .setManifestKeyPrefix('build/')

    /*
     * ENTRY CONFIG
     *
     * Add 1 entry for each "page" of your app
     * (Including one that's included on every page - e.g. "app")
     *
     * Each entry will result in one JavaScript file (e.g. app.js)
     * and one CSS file (e.g. app.css) if your JavaScript imports CSS.
     */
    .addEntry('app', './assets/app.js')
    .addEntry('input-focus', './assets/styles/js/input-focus.js')
    .addEntry('modal', './assets/styles/js/modal.js')
    .addEntry('modal_description', './assets/styles/js/modal_description.js')
    .addEntry('panier', './assets/styles/js/panier.js')
    .addEntry('addPanier', './assets/styles/js/addPanier.js')
    .addEntry('showBackhugs', './assets/styles/js/showBackhugs.js')
    .addEntry('vanilla-tilt', './assets/styles/js/vanilla-tilt.min.js')
    .addEntry('vanilla-tilt-init', './assets/styles/js/vanilla-tilt-init.js')
    .addEntry('bootstrap', './assets/styles/bootstrap/js/bootstrap.bundle.min.js')
    .addEntry('jquery', './assets/styles/jquery/jquery.min.js')

    // When enabled, Webpack "splits" your files into smaller pieces for greater optimization.
    .splitEntryChunks()

    // will require an extra script tag for runtime.js
    // but, you probably want this, unless you're building a single-page app
```

## API

Les API ont été réalisé manuellement via le controller '*ApiController.php*' dans le cas où vous voudriez ajouter des API vous pouvez utiliser la librairie '*API Platform*' qui est installé en version 2.5. Il vous suffira donc d'ajouter les annotations dans les entity.

```
class ApiController extends AbstractController {
    /**
     * @Route("/api/products", name="api_products")
     */
    public function productsApi(ProductRepository $productRepository): Response {
        $products = $productRepository->findAll();

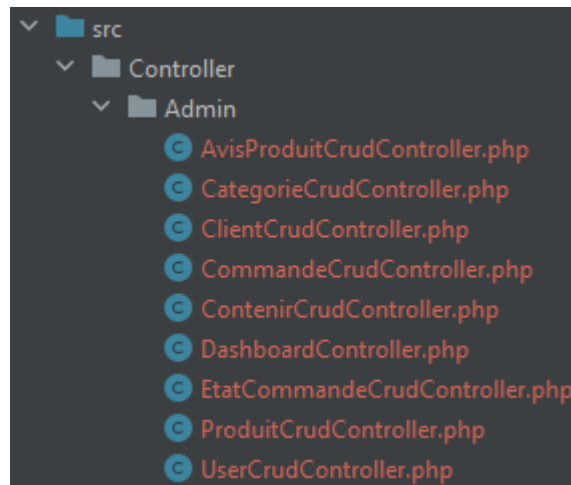
        //region Serialiser pour produit avec caractéristiques et références
        $encoder = new JsonEncoder();
        $defaultContext = [];
        $abstractNormalizer = (AbstractNormalizer::class) => function ($object, $format, $context) {
            return $object->getLibelle();
        };
        $normalizer = new ObjectNormalizer($abstractNormalizer, null, $encoder, null, $propertyAccessor, null, $objectTypeResolver, null, $phpDocumentorFactory, null, $httpCacheFactory, null, $defaultContext);
        $serializer = new Serializer([$normalizer], [$encoder]);
        //end-region

        // Suppression des paramètres dans libelles avec l'appel mobile (variance de la réponse)
        foreach ($products as $product) {
            $product->setCaracteristiques(new ArrayCollection());
        }

        echo "après";
        echo($serializer->serialize($products, 'json'));
        echo "</pre>";
        $response = new Response();
        $response->setContent($serializer->serialize($products, 'json'));
        $response->headers->set('Content-Type', 'application/json');
        return $response;
    }
}
```

## Back Office

Le back office est réalisé avec Easyadmin en version 3.2 et permet de gérer le contenu du site.



```
/**
 * @IsGranted("ROLE_ADMIN")
 */
class ProduitCrudController extends AbstractCrudController
{
    public static function getEntityFqcn(): string
    {
        return Produit::class;
    }

    /**
     * public function configureCrud(Crud $crud): Crud
     * {
     *     return $crud->
     *     ;
     * }

    public function configureFields(string $pageName): iterable
    {
        return [
            ImageField::new( propertyName: 'lienImage')
                ->setBasePath( path: '../Images/Produits')
                ->setUploadDir( uploadDirPath: 'Images/Produits')
                ->setUploadedFileNamePattern( patternOrCallable: '[year]-[month]-[day]_[name].[extension]'),
            TextField::new( propertyName: 'Marque'),
            TextField::new( propertyName: 'Libelle'),
            NumberField::new( propertyName: 'Tarif'),
            IntegerField::new( propertyName: 'Stock'),
            AssociationField::new( propertyName: 'id_categorie'),
        ];
    }
}
```

## Services de paiement

Le service de paiement est réalisé avec l'API Paypal.

```
<div id="btn-paypal-container" class="offset-lg-3 col-lg-3 col-sm-12"> </div>
</div>
</form>

<script src="https://www.paypal.com/sdk/js?client-id=Aia7P3SSeM2yG2Lisc4VPBqWUDeRBIvgdWYradMFo5e8cidnrcEfaQ17-Z167rFvE4ype2Ap0HwHdQcuxponents=buttons"></script>
<script>
  var commande = null;
  paypal.Buttons({
    style: {
      layout: 'vertical',
      color: 'black',
      shape: 'rect',
      label: 'paypal'
    },
    createOrder: function(data, actions) {...},
    onApprove: function(data, actions) {...}
  }).render('#btn-paypal-container');
</script>
```



# Style

## Card Produit

Les card des produits ont été réalisé avec la librairie [vanilla-tilt.js](https://dmitrybaranovskiy.github.io/tilt.js/) afin de donner un aspect 3D à ces éléments.

La librairie n'est pas importée avec un lien cdn mais bien installé dans les assets, sous le libelle '*vanilla-tilt.min.js*', de même que le fichier d'initialisation '*vanilla-tilt-init.js*' des éléments impactés par cette librairie

L'aspect des card est défini dans le fichier cardProduit.css situé dans les assets.

## Graphiques

Les graphiques sont réalisé avec la librairie JavaScript [Chart.js](https://dmitrybaranovskiy.github.io/chart.js/) cette dernière permet la génération simplifié de graphique. Cette dernière est importé via un lien cdn et l'ensemble des scripts d'initialisation sont localisé dans la Template '*dashboard.html.twig*', Template d'accueil du back office et plus précisément dans les fonctions success des requêtes ajax des données des graphiques.

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script type="text/javascript">

    var months = ['Janvier', 'Fevrier', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Août', 'Septembre', 'Octobre', 'Novembre', 'Decembre'];
    let width, height, gradient;
    function getGradient(ctx, chartArea) {...}

    function createRadialGradient3(context, c1, c2, c3) {...}

    $.post('{{ path('/getUserNumberJson') }}', function (data){...});
    $.post('{{ path('/getCommandeClient') }}', function (data){...});
    $.post('{{ path('/getProduitByCategorie') }}', function (data){...});
    $.post('{{ path('/getProduitStockLow') }}', function (data){...});
    $.post('{{ path('/getCommandeJson') }}', function (data){...});

</script>
```