ELECTRIKA Gestion

Projet de fin de formation – Titre Professionnel DWWM

Réalisé par : Marc Longmar  
Formation : Développeur Web et Web Mobile (AFPA Cannes)  
Période : 15 mars 2025 – 1er juin 2025

1. Présentation générale du projet

# Sommaire

1. Présentation générale du projet.................................... Page 2

2. Environnement de travail et configuration.......................... Page 3

3. Maquettage et accessibilité........................................ Page 4

4. Développement Front-End............................................ Page 5

5. Développement Back-End............................................. Page 6

6. Composants métier orientés objet................................... Page 7

7. Déploiement et veille technologique................................ Page 8

8. Bilan personnel et compétences acquises............................ Page 9

1. Présentation générale du projet

1.1 Intitulé et présentation

Le projet présenté s’intitule « ELECTRIKA Gestion ». Il s’agit d’une application web destinée à suivre en temps réel la disponibilité du matériel dans le cadre d’activités événementielles. Cette solution vise à optimiser la gestion logistique des équipements techniques en apportant un outil accessible et automatisé aux techniciens et responsables d’exploitation.  
Le projet a été mené entre le 15 mars 2025 et le 1er juin 2025, dans le cadre de la formation Développeur Web et Web Mobile (DWWM) suivie à l’AFPA de Cannes.

1.2 Objectifs

- Suivre l’état du matériel loué ou utilisé sur les événements à venir, jusqu’à 4 semaines glissantes ;  
- Permettre à chaque technicien d’assumer le rôle de préparateur de commande de façon autonome ;  
- Générer automatiquement des documents PDF (bons de préparation, de livraison et de retour) ;  
- Informer le responsable technique par mail à chaque étape importante du processus ;  
- Offrir une interface claire et responsive accessible depuis tout support (ordinateur, mobile, tablette).

1.3 Contexte

Ce projet est né d’un constat personnel, issu de 20 années d’expérience dans le domaine de l’événementiel. Dans de nombreuses situations, le suivi du matériel était chronophage, peu fiable ou dépendait de trop peu de personnes. Pour répondre à ce besoin, l’idée a été de développer une solution numérique simple, permettant à tous les techniciens de gérer eux-mêmes leurs besoins logistiques.  
Bien que développé en dehors du cadre d’une entreprise réelle, ce projet s’inscrit dans une logique professionnelle concrète. Il a été entièrement réalisé en autonomie , sur mon temps personnel, avec un fort engagement sur la qualité et la rigueur. Le projet a pour ambition de servir de base à une solution réelle, qui pourrait être adaptée par l’entreprise ELECTRIKA ou d’autres acteurs du secteur.

2. Environnement de travail et configuration

2.1 Outils de développement utilisés

L’environnement de développement a été mis en place dès le début du projet afin d’assurer une productivité optimale et un bon niveau de rigueur dans la gestion du code et des versions.  
Les outils principaux utilisés sont :  
- Visual Studio Code (VSCode) pour l’écriture du code source, avec extensions dédiées à Symfony et PHP ;  
- WampServer pour le serveur local Apache et MySQL ;  
- Node.js et NPM pour la gestion des dépendances front-end ;  
- Symfony CLI pour créer, configurer et lancer le projet ;  
- Composer pour la gestion des dépendances PHP ;  
- phpMyAdmin pour la visualisation de la base de données ;  
- MySQL Workbench pour modéliser le MCD et MPD ;  
- MailDev pour la simulation d’envois d’e-mails pendant le développement.

2.2 Gestion de versions et collaboration

Le projet est versionné sur Git et hébergé sur une plateforme GitHub. Cela a permis de suivre les évolutions du projet, de sécuriser les sauvegardes et de revenir en arrière si nécessaire.  
Bien que développé seul, le respect des conventions de commits, de branches et de structure de dépôt a été appliqué.

2.3 Conteneurs et services intégrés

Le projet s’appuie sur plusieurs services et bibliothèques qui assurent le bon fonctionnement de l’environnement de développement :  
- Symfony (framework MVC) ;  
- Doctrine ORM pour la gestion des entités et de la base de données ;  
- Webpack Encore pour le traitement des fichiers CSS/JS/SCSS ;  
- DomPDF pour la génération de bons de préparation/livraison/retour ;  
- Bootstrap pour le responsive design ;  
- JavaScript natif pour la logique côté client.

2.4 Documentation et veille technologique

Tous les outils et technologies ont été documentés, en français et en anglais, grâce aux documentations officielles des éditeurs et des communautés (Symfony, PHP.net, MDN Web Docs, GitHub).  
Une veille technique a été réalisée en continu sur les sujets suivants : performances Symfony, mise à jour des dépendances, sécurité des formulaires, et retours d’expérience sur les bonnes pratiques DevOps.  
Cette veille m’a permis de mieux comprendre les problématiques de sécurité et d’organisation dans un contexte de déploiement et de production.

3. Maquettage et accessibilité

3.1 Outils de conception utilisés

La conception des interfaces utilisateur a été réalisée à l’aide de l’outil Figma. Celui-ci a permis de créer des maquettes claires, cohérentes avec l’identité visuelle du projet et faciles à adapter pour les vues desktop et mobile. L’usage de Figma a facilité la prévisualisation des enchaînements de pages, la structuration des zones fonctionnelles, et la définition des gabarits répétables.

3.2 Respect de la charte graphique

Une attention particulière a été portée à l’uniformité visuelle de l’application : typographies harmonisées, code couleur stable, et composants Bootstrap réutilisés. La charte graphique sobre mais professionnelle permet une lecture claire, même en conditions de lumière réduite sur les lieux d’intervention.

3.3 Accessibilité et expérience utilisateur

Le projet a intégré les principes d’accessibilité numérique tout au long du développement. Les éléments suivants ont été pris en compte :  
- Ajout de balises ALT sur les images ;  
- Titres hiérarchisés avec balises H1 > H2 > H3 ;  
- Utilisation des attributs ARIA (aria-label, role) sur les composants interactifs ;  
- Contraste suffisant entre le fond et le texte ;  
- Formulaires avec labels explicites et placeholders pertinents.

3.4 Adaptabilité mobile

L’application a été pensée dès le départ pour s’adapter aux différents supports (mobile, tablette, desktop) selon la méthode 'Mobile First'. L’utilisation de Bootstrap, combinée à des media queries personnalisées, permet une mise en page fluide et adaptée :  
- Utilisation des unités relatives (%, rem, em) ;  
- Meta viewport configurée ;  
- Réduction ou transformation des composants sur les petits écrans.

3.5 Schéma d’enchaînement des pages

Un schéma de navigation a été produit sur Figma pour formaliser le parcours utilisateur :  
Accueil → Planning → Liste des événements → Fiche événement → État (préparation / livraison / retour).  
Ce cheminement est fluide et permet d’accéder en un minimum de clics aux informations essentielles. Une interface de type tableau de bord est envisagée pour la prochaine version.

4. Développement Front-End

4.1 Technologies utilisées

Le développement de l’interface utilisateur s’est appuyé principalement sur le moteur de templates Twig, fourni par Symfony. Twig permet de générer dynamiquement les vues à partir des données serveur, tout en assurant une bonne séparation entre la logique métier et la présentation.  
Bootstrap a été utilisé pour garantir une mise en page responsive et cohérente sur tous les supports (ordinateur, mobile, tablette). Des composants réutilisables ont été personnalisés à l’aide de SCSS pour adapter l’esthétique générale aux contraintes du projet.

4.2 Intégration statique

La structure HTML/CSS respecte les bonnes pratiques du W3C avec une hiérarchie de balises claire, des sections sémantiques bien identifiées et des classes CSS structurées. Les pages ont été codées selon les maquettes Figma validées, avec une attention particulière portée à la lisibilité, au contraste et à l’adaptabilité des composants.  
L’ensemble des contenus a été intégré dans des composants Twig réutilisables pour favoriser la maintenance du code.

4.3 Dynamisme de l’interface

Le projet intègre plusieurs éléments dynamiques en JavaScript natif. Par exemple, lors de la sélection d’un événement, seuls les articles associés sont affichés dynamiquement sans rechargement de page, en utilisant la logique d’affichage conditionnel avec des classes CSS `d-none`.  
Cette logique améliore l’expérience utilisateur, tout en restant simple à maintenir.

4.4 Sécurisation et accessibilité

L’affichage dynamique respecte les standards d’accessibilité en évitant les pièges classiques : textes alternatifs, bonne structure HTML, et navigation clavier. Du côté sécurité, Twig protège nativement contre les attaques XSS en encodant automatiquement les variables affichées. Cela garantit que même les entrées utilisateur affichées ne peuvent pas compromettre l’intégrité du site.

4.5 Référencement et bonnes pratiques

Même si le projet n’a pas vocation immédiate à être indexé, les balises meta, les titres hiérarchisés, l’usage de `robots.txt` et `sitemap.xml` ont été intégrés en prévision d’une mise en ligne publique. Le code est organisé pour respecter la structure recommandée par la W3C et permettre un bon positionnement dans les moteurs de recherche à long terme.

5. Développement Back-End

5.1 Base de données relationnelle

Le projet repose sur une base de données relationnelle conçue à partir d’un MCD (Modèle Conceptuel de Données) et traduite en MPD (Modèle Physique de Données) via MySQL Workbench. Elle comprend 11 tables principales (hors tables techniques Doctrine).  
Les entités couvrent les éléments suivants : utilisateurs, produits, événements, contacts, sites, détails d’événements, galeries, catégories, etc.  
Les contraintes d’intégrité (PK, FK, UNIQUE, NOT NULL) ont été respectées pour garantir la fiabilité des relations.

5.2 Utilisation de Doctrine ORM

Doctrine a été utilisé comme outil principal de gestion de la base. Il permet une abstraction des requêtes SQL grâce à la manipulation d'entités PHP. Les migrations ont permis de maintenir à jour le schéma sans perdre les données existantes.  
Des méthodes personnalisées ont été intégrées aux repositories, comme :  
- `findByEventStatus()`  
- `countStockByProduct()`  
- `countOrdersByStatus()`

5.3 Développement des composants métier

Les services Symfony ont permis de gérer des fonctionnalités spécifiques telles que :  
- La génération de bons au format PDF (DOMPDF) ;  
- L’envoi automatique d’e-mails à chaque changement d’état.  
Ces services sont appelés à chaque transition de statut d’un événement (new, prêt, livré, retour).

5.4 Sécurité et validation des données

Le système de rôles différencie `ROLE\_ADMIN` et `ROLE\_TECH`. Les routes sensibles sont protégées via les annotations Symfony.  
Les mots de passe sont hashés avec le composant Security de Symfony. Des tokens CSRF sont appliqués aux formulaires sensibles.  
Les entrées utilisateur sont systématiquement validées avant traitement serveur.

5.5 Jeux d’essai et tests fonctionnels

Des données de test ont été initialement générées via DataFixtures. Par la suite, les insertions ont été réalisées manuellement afin de mieux contrôler les droits et rôles. Les tests manuels ont porté sur les :  
- Créations d’événements, produits, utilisateurs ;  
- Génération de PDF ;  
- Notifications e-mail ;  
- Calcul des stocks selon les sorties prévues.

6. Composants métier orientés objet

6.1 Fonctionnalités couvertes

Le cœur de la logique métier repose sur la gestion des événements et de leur cycle de vie. À chaque état d’un événement (nouveau, prêt, livré, retour), une série de traitements sont lancés : mise à jour des quantités disponibles, génération de documents, notifications par e-mail, etc.  
Ces traitements ont été isolés dans une couche service dédiée, ce qui permet de mieux organiser le code et de faciliter son évolution future.

6.2 Programmation orientée objet (POO)

L’ensemble du projet respecte les principes de la programmation orientée objet :  
- Création de classes métiers dédiées (PDFGeneratorService, EventMailService, etc.) ;  
- Utilisation d’interfaces et d’abstractions pour faciliter les tests et la réutilisation ;  
- Encapsulation des règles métier dans des méthodes claires ;  
- Mise en place de constructeurs, héritage et namespaces pour organiser proprement les services.

6.3 Nommage et documentation

Le projet suit une convention de nommage en PascalCase pour les classes et camelCase pour les méthodes. Tous les noms sont rédigés en anglais, dans un souci de lisibilité et de respect des standards du développement.  
Le code source est commenté sur les parties complexes et les services sont documentés à l’aide de blocs /\*\* \*/ pour faciliter leur compréhension par un tiers.

6.4 Vérifications et jeux d’essai

Les services métiers ont été testés à l’aide de jeux d’essai concrets (fixtures et données manuelles). L’enchaînement des traitements critiques (modification de stock, retour, édition PDF) a été testé en condition réelle avec différents profils utilisateurs. L’ensemble des cas critiques a été vérifié pour assurer la fiabilité métier.

7. Déploiement et veille technologique

7.1 Mise en production

Le projet a été déployé sur un hébergement mutualisé OVH. La mise en production a été automatisée à l’aide d’un script `deploy.sh` contenant les commandes suivantes :  
- `git pull` pour récupérer les mises à jour ;  
- `composer install` ou `update` pour les dépendances PHP ;  
- `npm install` et `npm run build` pour les assets ;  
- `php bin/console doctrine:migrations:migrate` pour mettre à jour la base de données.  
Des vérifications post-déploiement ont été réalisées sur la disponibilité, la structure des pages, et l’envoi de mails (via Maildev en local, SMTP en production).

7.2 Scripts utilisés

Un script Bash nommé `deploy.sh` a été créé pour automatiser les étapes du déploiement. Il garantit une mise à jour rapide et sécurisée de l’application lors de l’évolution du projet. Ce script est documenté ligne par ligne pour permettre à un futur développeur de s’en servir sans difficulté.

7.3 Suivi des évolutions techniques

Une veille continue a été menée sur plusieurs axes :  
- Évolutions du framework Symfony (newsletter, changelog GitHub) ;  
- Bonnes pratiques de sécurité (OWASP, blogs développeurs) ;  
- Performances de déploiement (lecture de retours sur Dev.to, Medium).  
Cette veille a permis de repérer certaines failles potentielles (CSRF, injections SQL) et d’anticiper la nécessité de migrations futures. Un audit de sécurité est prévu pour la version 2 du projet, en incluant des tests avec outils automatisés.

8. Bilan personnel et compétences acquises

8.1 Compétences techniques développées

Ce projet m’a permis de consolider de nombreuses compétences techniques essentielles à la pratique du métier de développeur web. Parmi celles-ci :  
- La maîtrise du framework Symfony, de ses services et de son système de routage ;  
- L’utilisation avancée de Doctrine ORM pour gérer des bases relationnelles complexes ;  
- La structuration d’un projet front-end avec Bootstrap, JavaScript et Twig ;  
- La mise en place de tests, de fixtures et de sécurisation des formulaires avec CSRF et validation.  
L’ensemble du code a été documenté, maintenable et conforme aux standards.

8.2 Méthodologie et gestion du projet

Travailler en autonomie sur un projet aussi complet m’a obligé à structurer mon travail, à planifier des étapes claires et à apprendre à prioriser. Inspiré de la méthode agile, j’ai découpé mon projet en sprints hebdomadaires : maquettage, développement front, back-end, tests, déploiement, documentation.  
J’ai appris à corriger des erreurs complexes grâce à la veille, au soutien de la communauté, et à des outils de débogage comme Maildev et Web Profiler.

8.3 Obstacles surmontés

Les principales difficultés rencontrées concernaient la gestion des relations complexes entre entités (ManyToMany, cascade persist), ainsi que la logique métier liée aux états d’un événement.  
Le déploiement sur un hébergement mutualisé a également nécessité une adaptation manuelle et un ajustement des scripts et permissions.

8.4 Évolution et prochaines étapes

Ce projet m’a motivé à continuer vers une spécialisation back-end. La prochaine étape serait de transformer cette base fonctionnelle en application réellement utilisée par l’entreprise, avec une couche mobile plus poussée, un tableau de bord commercial et un accès client.  
Je me sens désormais prêt à intégrer une entreprise, à évoluer dans un cadre en équipe, et à relever des projets plus ambitieux.