

Dossier du projet : Création du site du camping Le Maine Blanc



django

SOMMAIRE

1. Introduction
2. Liste des compétences mobilisées
3. Résumé du projet
4. Cahier des charges et Expression des besoins
 - 4.1. Contexte du projet
 - 4.2. Objectifs du projet
 - 4.3. Public cible
 - 4.4. Fonctionnalités attendues
 - 4.5. Contraintes et exigences
 - 4.6. Livrables attendus
5. Spécifications techniques du projet
 - 5.1. Choix techniques et environnement de développement
 - 5.2. Architecture du projet
 - 5.3. Sécurité et gestion des données
 - 5.4. Web mobile et responsive design
 - 5.5. Optimisation SEO et performance
 - 5.6. Multilinguisme et accessibilité
6. Réalisations du projet
 - 6.1. Présentation générale
 - 6.2. Extraits de code et commentaires
 - 6.2.1. Formulaire de réservation

- 6.2.2. Calcul du prix et gestion des acomptes
- 6.2.3. Vue de réservation sécurisée
- 6.2.4. Templates responsives (Bootstrap 5)
- 6.3. Sécurité et bonnes pratiques
 - 6.3.1. Sécurité des données et des formulaires
 - 6.3.2. Sécurisation des communications et de l'environnement
 - 6.3.3. Envoi d'e-mails sécurisé
 - 6.3.4. Bonnes pratiques de développement
 - 6.3.5. Sauvegarde et intégrité de la base de données
- 6.4. Points forts techniques
- 7. Jeu d'essai fonctionnel
 - 7.1. Objectif du jeu d'essai
 - 7.2. Scénarios de tests principaux
 - 7.3. Résumé des tests
 - 7.4. Améliorations et évolutions techniques envisagées
- 8. Veille sur les vulnérabilités de sécurité
 - 8.1. Sources de veille et outils utilisés
 - 8.2. Vulnérabilités identifiées récemment et mesures appliquées
 - 8.2.1. Injection SQL
 - 8.2.2. Cross-Site Scripting (XSS)
 - 8.2.3. Cross-Site Request Forgery (CSRF)
 - 8.2.4. Exposition des e-mails et données personnelles
 - 8.2.5. Gestion des mots de passe et accès admin
 - 8.3. Bonnes pratiques mises en œuvre
 - 8.4. Plan de veille et mise à jour continue

9. Situation de recherche sur un site anglophone
 - 9.1. Besoin d'information
 - 9.2. Méthode de recherche
 - 9.3. Synthèse des informations retenues (appliquées au projet)
10. Extrait du site anglophone et traduction
 - 10.1. Extrait (anglais) : source OWASP Django Security Cheat Sheet
 - 10.2. Traduction de l'extrait en français
11. Conclusion
12. Annexes
 - 12.1. Arborescence du projet Django
 - 12.2. Technologies utilisées
 - 12.3. Exemples d'interfaces
 - 12.4. Liens utiles
 - 12.5. Glossaire

1. Introduction

Ce dossier présente la conception, le développement et la documentation technique complète du site web du Camping Le Maine Blanc, réalisé dans le cadre de mon projet professionnel de fin de formation en développement web et mobile.

2. Liste des compétences mobilisées

Le projet « Création du site web du Camping Le Maine Blanc » a permis de mobiliser et mettre en œuvre plusieurs compétences du référentiel du Titre Professionnel Développeur Web et Web Mobile (DWWM).

Ces compétences se répartissent dans les deux activités principales du titre :

- Bloc 1 : développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité,
- Bloc 2 : développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

Compétence	Description	Mise en œuvre dans le projet
1.1. Maquetter une application	Concevoir les interfaces et parcours utilisateurs avant le développement.	Réalisation des maquettes et prototypes sur Figma (versions desktop, tablette et mobile) à partir du Design System défini.
1.2. Réaliser une interface utilisateur web statique et adaptable	Intégrer le HTML/CSS, gérer la responsivité et la cohérence graphique.	Intégration des pages informatives avec HTML5, CSS3 et Bootstrap 5 pour garantir un affichage optimal sur tous les écrans.
1.3. Développer une interface utilisateur web dynamique	Ajouter de l'interactivité avec JavaScript.	Mise en place de comportements dynamiques : menu responsive, validation côté client, interaction sur les formulaires.
1.4. Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce	Utiliser un CMS ou un framework pour le front-end.	Utilisation du framework Django pour la gestion du contenu et des pages dynamiques, avec administration intégrée.
1.5. Mettre en œuvre des recommandations de sécurité dans une application web	Sécuriser les échanges et les données utilisateurs.	Protection CSRF, validation serveur, gestion des sessions et formulaires sécurisés avec Django.

Compétence	Description	Mise en œuvre dans le projet
2.1. Créer une base de données	Concevoir et structurer les données.	Modélisation de la base SQLite3 avec les entités Booking, Price, MobileHome, etc.
2.2. Développer les composants d'accès aux données	Gérer les requêtes CRUD et la logique métier.	Création des modèles Django, gestion des réservations, calcul des prix et vérification des disponibilités.
2.3. Développer les composants métier côté serveur	Implémenter la logique de traitement et de calcul.	Calcul automatique du prix total et de l'acompte, gestion des périodes tarifaires et des suppléments.
2.4. Documenter le déploiement d'une application dynamique	Préparer la mise en production.	Documentation de déploiement sur Render, configuration du fichier requirements.txt, et gestion des variables d'environnement.
2.5. Mettre en œuvre des recommandations de sécurité dans une application web	Protéger les données et limiter les vulnérabilités.	Gestion des formulaires sécurisés, validation côté serveur, contrôle d'accès à l'administration Django, HTTPS en production.
2.6. Concevoir et mettre en œuvre des tests unitaires	Vérifier la fiabilité du code et la cohérence des fonctionnalités.	Utilisation de Pytest et création de tests unitaires pour les formulaires, les modèles et les vues.

Ce projet m'a permis de mobiliser l'ensemble des compétences clés du référentiel DWWM :

- Conception d'une application web complète,
- Développement front-end et back-end,
- Intégration des bonnes pratiques de sécurité,
- Mise en œuvre d'une architecture MVC avec Django,
- Test et déploiement d'une application dynamique,
- Gestion de la responsivité et de l'expérience utilisateur sur web et mobile.

3. Résumé du projet

Le projet « Création du site web du Camping Le Maine Blanc » s'inscrit dans le cadre de la formation au titre professionnel Développeur Web et Web Mobile. Il a pour objectif de moderniser un ancien site obsolète, peu adapté aux usages actuels et de proposer une solution complète de présentation et de réservation en ligne.

La site vise à offrir une expérience utilisateur fluide, responsive et multilingue afin de répondre aux besoins d'une clientèle variée et internationale. Il permet de consulter les informations du camping, de découvrir les hébergements et d'effectuer une réservation ou une demande de contact en ligne.

Le projet a été développé avec le framework Django (python) pour le back-end et Bootstrap 5, HTML5, CSS3 et JavaScript pour le front-end. La base de données SQLite3 gère les réservations, les tarifs, les suppléments et les disponibilités. L'interface d'administration Django permet aux gérants du camping de gérer le contenu du site et les réservations en toute autonomie.

Le site intègre des mesures de sécurité (protection CSRF, validation des formulaires, gestion des sessions, contrôle d'accès) et respecte les bonnes pratiques du web mobile. Il a été testé avec Pytest et déployé sur la plateforme Render.

Ce projet constitue une application complète et fonctionnelle, alliant design, performance et sécurité, représentative des compétences acquises tout au long de ma formation.

4. Cahier des charges et expression des besoins

4.1. Contexte du projet

L'objectif du projet « Camping Le Maine Blanc » était de concevoir un site web professionnel et moderne pour un camping familial situé en Gironde afin d'améliorer sa visibilité en ligne et de faciliter les réservations.

Le camping disposait auparavant d'un site vitrine peu ergonomique, non adapté aux appareils mobiles et ne disposant pas de système de réservation en ligne.

Le nouveau site devait offrir une meilleure expérience utilisateur, optimiser le référencement naturel (SEO) et répondre aux standards actuels du web responsive et multilingue.

4.2. Objectifs du projet

Le site doit répondre à plusieurs objectifs :

- Moderniser l'image du camping à travers un site attractif et fluide,
- Faciliter l'accès à l'information pour les visiteurs,
- Permettre la réservations d'emplacements en ligne et l'envoi de demande de contact aux gérants,
- Proposer une navigation claire et intuitive, y compris sur mobile,
- Améliorer la visibilité sur les moteurs de recherche (SEO),
- Proposer un site multilingue (français, anglais, espagnol, allemand et néerlandais)

4.3. Public cible

Le site s'adresse principalement à :

- Des familles et couples recherchant un hébergement de vacances en Gironde,
- Des touristes étrangers, notamment européens, grâce à la version multilingue,
- Des ouvriers en mission temporaire dans la région,
- Le gérant du camping pour la mise à jour et la consultation des demandes de réservation.

4.4. Fonctionnalités attendues

Le cahier des charges prévoit les fonctionnalités suivantes :

- Page Accueil et A Propos présentant le camping et ses atouts visuels,
- Page Hébergements présentant les types d'emplacements et mobil-homes disponibles,
- Formulaire de contact et de réservation avec validation des champs, messages et e-mails de confirmation,
- Pages dédiées aux informations pratiques (horaires, tarifs), aux services et aux activités touristiques,
- Gestion multilingue avec sélecteur de langue,
- Version responsive adaptée à tous les écrans,
- Optimisation SEO pour un meilleur positionnement sur Google,
- Lien vers les réseaux sociaux et Google Maps,
- Formulaire conforme au RGPD (consentement à la collecte des données).

4.5. Contraintes et exigences

Les contraintes et exigences à respecter étaient :

- Le site devait être entièrement fonctionnel et responsive, compatible avec les principaux navigateurs (Chrome, Edge, Firefox),
- Le développement devait s'appuyer sur des technologies modernes assurant la stabilité, la sécurité et la maintenabilité du site,
- La sécurité devait être garantie via la gestion des formulaires, la validation des entrées et la protection CSRF,

- Le projet devait être hébergé en ligne pour la version de démonstration destinée aux gérants et au jury (via Render),
- Le code devait être versionné sur GitHub,
- Les délais de réalisation étaient définis dans le cadre de la formation, avec des livrables progressifs (maquettes, prototypes, version finale).

4.6. Livrables attendus

Les livrables attendus dans le cadre du projet étaient :

- Le site web finalisé et déployé en ligne,
- Le code source complet hébergé sur GitHub,
- Les captures d'écran et démonstrations illustrant les fonctionnalités principales,
- La documentation technique incluant le fichier README, le guide d'installation et d'utilisation.

Ces livrables répondent à la fois aux objectifs pédagogiques du projet et aux attentes réelles du site pour les gérants du camping.

5. Spécifications techniques du projet

5.1. Choix techniques et environnement de développement

Le site Camping Le Maine Blanc a été développé avec le framework Django (Python), choisi pour sa robustesse, sa sécurité native et sa structure modulaire facilitant la maintenance.

L'interface utilisateur a été conçue avec HTML5, CSS3, Bootstrap 5 et JavaScript permettant un rendu moderne et responsive.

L'environnement de développement reposait sur :

- IDE : Visual Studio Code,
- Framework backend : Django 5,
- Langages : Python, HTML, CSS, JavaScript,
- Framework frontend : Bootstrap 5,
- Base de données : SQLite3 (en phase de développement),
- Versioning : Git et GitHub,
- Maquettage : Figma,
- Hébergement : Render (pour la mise en ligne de la version de démonstration).

Ces choix techniques ont été faits dans le but d'obtenir un site sécurisé, évolutif et performant, répondant aux critères du titre professionnel DWWM et aux besoins concrets du camping.

5.2. Architecture du projet

Le projet est structuré selon l'architecture MVC (Modèle - Vue - Contrôleur) proposée par Django :

- Le modèle gère la structure des données (modèles Reservation, Booking, Price, ...),
- Les vues contiennent la logique applicative et renvoient les informations aux templates,
- Les templates (HTML + Bootstrap) assure la présentation et l'ergonomie du site.

Arborescence simplifiée :

```
maineb Blanc_project/  
|  
|--- bookings          # Application de réservation en ligne  
|   |--- templates      # Fichiers HTML  
|   |--- tests          # Tests models, views et forms  
|   |--- models.py      # Modèles de données  
|   |--- views.py       # Logique de traitement  
|   |--- urls.py        # Routage  
|   |--- forms.py       # Gestion des formulaires  
|--- core              # Application principale  
|--- docs              # Gestion de la documentation  
|--- locale            # Gestion des traductions  
|--- maineb Blanc_project # Fichiers de configuration Django  
|--- reservations      # Application de contact  
|--- static            # Feuilles CSS, scripts JS, images  
|--- .env              # Variables de données  
|--- .gitignore        # Données non envoyées sur GitHub  
|--- db.sqlite3        # Base de données locales  
|--- manage.py         # Point d'entrée principal du projet  
|--- pytest.ini        # Initialisation des fichiers tests  
|--- README.md         # Fichier de documentation  
|--- requirements.txt   # Bibliothèques utilisées
```

L'organisation respecte les bonnes pratiques Django pour la lisibilité, la réutilisabilité et la maintenance du code.

5.3. Sécurité et gestion des données

Une attention particulière a été portée à la sécurisation des formulaires et des données :

- Utilisation de la protection CSRF intégrée dans Django,
- Validation stricte des champs de formulaire (type, longueur, format d'adresse mail, ...)
- Aucune donnée sensible n'est stockée sans vérification préalable,
- Gestion des erreurs et des redirections sécurisées,
- Respect du RGPD à travers un formulaire de consentement explicite,

- Configuration du fichier .env pour protéger les variables sensibles (clé secrète) lors du déploiement.

Ces mesures garantissent la fiabilité du traitement des réservations et la confidentialité des utilisateurs.

5.4. Web mobile et responsive design

L'ensemble du site a été conçu en mobile first, afin d'assurer une adaptation fluide sur tous les supports (ordinateurs, tablettes, smartphones).

Grâce à Bootstrap 5, les grilles flexibles et composants réactifs permettent :

- Une mise en page fluide avec colonnes et conteneurs adaptatifs,
- Une navigation claire et intuitive même sur petits écrans,
- Des images redimensionnées automatiquement selon le support,
- Des tests réalisés via les outils de développement Chrome et Firefox pour vérifier la compatibilité multi-écrans.

5.5. Optimisation SEO et performance

Pour garantir un bon référencement naturel (SEO), plusieurs optimisations ont été mises en place :

- Utilisation correcte des balises sémantiques HTML5 (<header>, <nav>, <main>, <section>, <footer>),
- Rédaction de titres et métadonnées uniques pour chaque page,
- Structure logique des balises <h1>, <h2>, <h3>, ...,
- Nommage cohérent des images avec balises ALT descriptives,
- Mise en cache du navigateur pour accélérer le chargement,
- Réduction du poids des images avant intégration,
- Tests de performance via Lighthouse (Google Chrome).

Ces optimisations contribuent à une meilleure visibilité du site sur les moteurs de recherche et à une expérience utilisateur fluide.

5.6. Multilinguisme et accessibilité

Le site a été conçu pour être multilingue : français, anglais, espagnol, allemand et néerlandais.

Cette fonctionnalité s'appuie sur le système de traduction intégré à Django (django.conf.locale).

Les fichiers .po et .mo contiennent les traductions de chaque texte affiché sur le site.

En parallèle, des efforts ont été faits pour améliorer l'accessibilité :

- Respect des contrastes de couleurs pour la lisibilité,
- Structure HTML sémantique compatible avec les lecteurs d'écran,
- Navigation simplifiée et intuitive pour tous les profils d'utilisateurs.

6. Réalisation du projet

6.1. Présentation générale

Cette partie décrit la réalisation complète du projet et met en avant :

- Les fonctionnalités clés développées : site multilingue, réservation en ligne, formulaire de contact, affichage des hébergements, gestion des disponibilités,
- Les extraits de code significatifs avec commentaires pour expliquer la logique,
- Les bonnes pratiques mises en œuvre pour la sécurité, la maintenabilité, le responsive design et la compatibilité mobile.

6.2. Extraits de code et commentaires

6.2.1. Formulaire de réservation

Le formulaire de réservation permet de saisir toutes les informations nécessaires à une réservation : type d'emplacement, dates, nombre d'occupants, options (électricité, dimensions), validation côté serveur pour la sécurité et la cohérence des données.

- Déclaration des champs principaux

```
booking_type = forms.ChoiceField(
    choices=[('', _('--- Choisissez ---'))] + BOOKING_TYPE_CHOICES_FOR_FORM,
    widget=forms.Select(attrs={'class': 'form-select'}),
    label=_("Type d'emplacement"),
    required=True
)

start_date = forms.DateField(
    widget=forms.DateInput(attrs={'type': 'date', 'min': timezone.localtime().isoformat()}),
    label=_("Date d'arrivée"),
    required=True
)
```

Ces champs permettent de choisir le type d'emplacement et la date d'arrivée. La date minimale est fixée au jour actuel pour éviter les réservations dans le passé.

- Validation des dates

```
# Date validation
if start_date < today:
    raise forms.ValidationError(_("La date d'arrivée ne peut pas être antérieure à aujourd'hui. "))
if start_date > end_date:
    raise forms.ValidationError(_("La date de départ doit être postérieure à la date d'arrivée. "))
```


On s'assure que les dates sont cohérentes et que la réservation n'est pas antérieure à aujourd'hui.

- Validation conditionnelle selon le type d'emplacement

```
if booking_subtype in ["caravan", "fourgon", "van", "camping_car"]:  
    if not cleaned_data.get("vehicle_length"):  
        self.add_error("vehicle_length", _("Le champ 'Longueur du véhicule' est obligatoire pour les caravanes et" \  
            " camping-cars."))  
  
if booking_subtype in ["tent", "car_tent"]:  
    if not cleaned_data.get("tent_width"):  
        self.add_error("tent_width", _("Le champ 'Largeur de la tente' est obligatoire pour les tentes."))  
    if not cleaned_data.get("tent_length"):  
        self.add_error("tent_length", _("Le champ 'Longueur de la tente' est obligatoire pour les tentes."))
```

Certains champs deviennent obligatoires selon le type d'emplacement choisi.

- Validation de l'électricité

```
if electricity == "yes":  
    cable_length = cleaned_data.get("cable_length")  
    if not cable_length:  
        self.add_error("cable_length", _("Le champ 'Longueur du câble' est obligatoire si l'électricité est incluse."))
```

Si l'utilisateur choisit l'option électricité, la longueur du câble devient obligatoire.

- Validation du nombre total de personnes

```
total_people = adults + children_over_8 + children_under_8  
  
if total_people > 6:  
    raise forms.ValidationError(  
        _("Le nombre maximum de personnes (adultes et enfants) ne peut pas dépasser 6."  
            " Pour toute demande particulière, merci de contacter directement le camping.")  
    )
```

On limite le nombre total de personnes par emplacement pour respecter les règles du camping.

En résumé, le formulaire est sécurisé et adaptatif, les champs changent selon le type d'emplacement choisi.

La cohérence des données et la sécurité côté serveur sont assurées.

Ces extraits représentent la logique principale et les contraintes métier du site.

6.2.2. Calcul du prix et gestion des acomptes

Cette partie du code gère le calcul automatique du prix total d'une réservation en fonction de plusieurs paramètres : le type d'emplacement (tente, caravane, camping-car, ...), la durée du séjour, la saison, la présence d'électricité, le nombre de personnes et les suppléments éventuels (animaux, véhicule supplémentaire, tente supplémentaire, ...).

Un acompte de 15% du montant total est ensuite calculé automatiquement pour confirmer la réservation.

- Détermination du nombre de nuits et du type d'emplacement

```
nights = max((self.end_date - self.start_date).days, 1)

subtype_to_type_map = {
    'tent': 'tent',
    'car_tent': 'tent',
    'caravan': 'caravan',
    'fourgon': 'caravan',
    'van': 'caravan',
    'camping_car': 'camping_car',
}

booking_type_for_price = subtype_to_type_map.get(self.booking_subtype, self.booking_type)
```

Le nombre de nuits est toujours au minimum d'une nuit.

Une correspondance (subtype_to_type_map) permet d'associer chaque sous-type de réservation général afin d'appliquer la bonne grille tarifaire.

- Récupération du tarif selon la saison et le type

```
season = self.get_season()
try:
    price = Price.objects.get(
        booking_type=booking_type_for_price,
        is_worker=False,
        season=season
    )
```

Le prix de base est récupéré dans le modèle Price, selon la saison et le type d'emplacement.

Le système est ainsi flexible et permet d'adapter automatiquement les tarifs selon la période (haute, moyenne ou basse).

- Application du tarif de base selon le type et l'électricité

```
if booking_type_for_price == 'camping_car':
    base_price = price.price_2_persons_with_electricity if electricity_yes else price.price_2_persons_without_electricity
    included_people = 2
else:
    included_people = 2 if self.adults >= 2 else 1
    if self.adults >= 2:
        base_price = price.price_2_persons_with_electricity if electricity_yes else price.price_2_persons_without_electricity
    else:
        base_price = price.price_1_person_with_electricity if electricity_yes else price.price_1_person_without_electricity
```

Le prix de base dépend du nombre d'adultes et de la présence d'électricité.

Cette logique garantit que le calcul soit juste et adapté à la situation du client.

- Ajout des suppléments (adultes supplémentaires, enfants, animaux, ...)

```
if supplement:
    total += extra_adults * (supplement.extra_adult_price or 0) * nights
    total += self.children_over_8 * (supplement.child_over_8_price or 0) * nights
    total += self.children_under_8 * (supplement.child_under_8_price or 0) * nights
    total += self.pets * (supplement.pet_price or 0) * nights
    total += self.extra_vehicle * (supplement.extra_vehicle_price or 0) * nights
    total += self.extra_tent * (supplement.extra_tent_price or 0) * nights
```

Chaque option ou supplément est ajouté au prix total selon le tarif journalier correspondant.

Le code reste modulaire : il est facile d'ajouter de nouveaux suppléments à l'avenir.

- Calcul de l'acompte

```
def calculate_deposit(self):
    """Calculate 15% deposit of total price, rounded to 2 decimals."""
    total_price = self.calculate_total_price()
    return round(total_price * Decimal('0.15'), 2)
```

Une fois le prix total calculé, un acompte de 15% est appliqué pour valider la réservation.

Ce pourcentage est clairement défini dans le code, ce qui permet une modification simple si la politique du camping évolue.

En résumé, le calcul du tarif est précis en fonction de plusieurs critères (type, saison, durée, options), la gestion des suppléments et des cas particuliers est automatiques, le calcul de l'acompte est clair et transparent.

Le code est structuré, maintenable et facilement adaptable à de futures évolutions tarifaires.

6.2.3. Vue de réservation sécurisée

Cette vue correspond à l'étape finale du processus de réservation.

Elle a pour rôle de valider les données saisies, enregistrer la réservation dans la base de données, envoyer les emails de confirmation (au client et à l'administrateur) et nettoyer la session pour éviter toute duplication.

L'objectif est de garantir un processus fiable, sécurisé et conforme aux bonnes pratiques de Django.

- Vérification des données et intégrité de la session

```
booking_data = request.session.get('booking_data')

if not booking_data:
    messages.error(request, _("Aucune donnée de réservation trouvée. Veuillez recommencer le processus de réservation."))
    return redirect('booking_form')

required_fields = ['first_name', 'last_name', 'address', 'postal_code', 'city', 'email', 'phone']
if not all(field in booking_data for field in required_fields):
    messages.error(request, _("Les informations de contact sont incomplètes. Veuillez compléter vos coordonnées."))
    return redirect('booking_details')
```

Avant toute création de réservation, la fonction vérifie que les données nécessaires sont présentes dans la session.

Cela empêche qu'un utilisateur contourne les étapes précédents ou envoie une requête incomplète directement.

- Conversion et nettoyage des données avant enregistrement

```
booking_data['start_date'] = date.fromisoformat(booking_data['start_date'])
booking_data['end_date'] = date.fromisoformat(booking_data['end_date'])
booking_data = {k: v for k, v in booking_data.items() if k in model_fields}
booking = Booking(**booking_data)
```

Les dates sont converties au bon format et seules les clés correspondant aux champs du modèle Booking sont conservées.

Cette étape protège contre l'injection de données non prévues et garantit la cohérence du modèle.

- Calcul automatique du prix et de l'acompte

```
supplement = SupplementPrice.objects.first()
total_price = booking.calculate_total_price(supplement=supplement)
deposit = booking.calculate_deposit()
booking.deposit_paid = True
booking.save()
```

La fonction appelle les méthodes précédemment définies (`calculate_total_price` et `calculate_deposit`) pour calculer le montant total du séjour et l'acompte de 15%.

Ces valeurs sont ensuite enregistrées dans la base de données.

- Envoi des emails de confirmation

```
email_admin = EmailMessage(
    subject=admin_subject,
    body=admin_message_final,
    from_email=settings.DEFAULT_FROM_EMAIL,
    to=[settings.ADMIN_EMAIL],
)
email_admin.content_subtype = "html"
email_admin.send(fail_silently=False)
```

Deux emails sont générés à partir de templates HTML : un email d'administration contenant tous les détails de la réservation, un email client dans la langue sélectionnée (grâce à `translation.override(request.LANGUAGE_CODE)`), confirmant la réservation et le montant de l'acompte.

Cette approche assure la communication multilingue et la traçabilité complète de chaque réservation.

- Nettoyage de la session et message utilisateur

```
if 'booking_data' in request.session:
    del request.session['booking_data']

messages.success(
    request,
    _("Merci ! Votre réservation a été confirmée. Un email de confirmation vous a été envoyé." if not is_render else
      "Merci ! Votre réservation a été confirmée. (Simulation d'envoi d'email sur Render.)")
)
```

Après la confirmation, toutes les données sont supprimées de la session afin d'éviter toute duplication si l'utilisateur recharge la page.

L'affichage d'un message de confirmation renforce l'expérience utilisateur.

Cette vue met en œuvre plusieurs bonnes pratiques : la validation stricte des données reçues avant tout enregistrement, la protection contre la réutilisation de session, le nettoyage des données en mémoire, la gestion multilingue (`translation.override`), l'envoi d'emails structurés et personnalisés, la gestion des erreurs d'envoi avec message utilisateur clair.

En résumé, cette vue constitue un point critique du projet : elle assure la sécurisation du processus de réservation et la fiabilité du stockage des données. Elle illustre la maîtrise des concepts essentiels de Django : sessions, ORM, validations, internationalisation et gestion des messages utilisateurs.

6.2.4. Templates responsive (Bootstrap 5)

L'architecture des templates du site repose sur un fichier principal `base.html`, qui définit la structure globale et les éléments communs à toutes les pages : l'en-tête (navigation), le pied de page, les métadonnées SEO et les blocs extensibles pour le contenu.

Ce modèle assure la cohérence visuelle et la compatibilité responsive sur tous les terminaux (smartphone, tablette, ordinateur).

- Structure principale

Le template `base.html` charge les bibliothèques nécessaires à l'affichage adaptatif :

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
```

Il intègre également un fichier CSS personnalisé :

```
<link rel="stylesheet" href="{% static 'css/style.css' %}?v={{ STATIC_VERSION }}" />
```

Ce fichier complète le style Bootstrap pour les besoins spécifiques du site (marges, couleurs, images, ...).

- En-tête responsive

La barre de navigation utilise les classes Bootstrap navbar, navbar-expand-lg et le composant collapse :

```
<nav class="navbar navbar-expand-lg fixed-top" aria-label="Main navigation">
  <div class="container-fluid">
    <!-- Logo -->
    <a class="navbar-brand" href="{% url 'home' %}">
      
    </a>

    <!-- Toggle button for mobile view -->
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
      aria-controls="navbarNav" aria-expanded="false" aria-label="{% trans 'Basculer la navigation'%}">
      <span class="navbar-toggler-icon"></span>
    </button>

    <!-- Menu -->
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ms-auto">...
    </ul>
    </div>
  </div>
</nav>
```

navbar-expand-lg permet au menu de s'afficher en pleine largeur sur les grands écrans et se replie sur mobile

navbar-toggler active le menu déroulant (collapse) pour une navigation fluide sur smartphone

Les classes *ms-auto* et *container-fluid* garantissent une mise en page flexible et centrée.

- Pied de page en grille Bootstrap

Le footer utilise le système de grilles responsives (row / col-md-*) pour adapter la disposition des blocs :

```

<footer class="pt-5 pb-3">
  <div class="container">
    <div class="row align-items-start">
      <!-- À propos -->
      <div class="col-md-4 mb-4">...
    </div>
      <!-- Infos pratiques -->
      <div class="col-md-5 mb-4">...
    </div>

      <!-- Réseaux sociaux -->
      <div class="col-md-3 mb-4">...
    </div>
    </div>

    <hr class="bg-light">

    <div class="text-center">
      <small>...
    </small>
    </div>
  </div>
</footer>

```

Sur mobile, les colonnes s'empilent automatiquement les unes sous les autres garantissant une lisibilité optimale.

- Blocs extensibles pour les pages enfants

Le template définit des blocs Django pour permettre à chaque page d'intégrer son propre contenu sans dupliquer la structure :

```

<!DOCTYPE html>
<html lang="{{ LANGUAGE_CODE|default:"fr" }}">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>{% block title %}{% trans "Accueil | Camping Le Maine Blanc" %}{% endblock %}</title>
  {% block styles %}{% endblock %}
</head>

<body>
  <header class="header">...
</header>
  <main>
    {% block content %}{% endblock %}
  </main>
  <footer class="pt-5 pb-3">...
</footer>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
  {% block extra_js %}{% endblock %}
</body>
</html>

```

Cela montre la réutilisabilité, la maintenabilité et la modularité du code.

Le code respecte les bonnes pratiques : le chargement des images est différé avec `loading= « lazy »`, les normes d'accessibilités sont respectées (balises `aria-label`, `alt`, navigation clavier), le sélecteur multilingue responsive est intégré via un menu déroulant (drapeaux + formulaire sécurisé), le footer et les menus sont adaptatifs grâce aux classes Bootstrap `col-md-*`, `collapse` et `dropdown`.

En résumé, l'ensemble du site conserve une mise en page claire, lisible et homogène quel que soit le support utilisé.

Les tests réalisés sur différents formats d'écran (mobile, tablette, desktop) ont confirmé une navigation fluide et sans débordement, des images et textes ajustés automatiquement, un temps de chargement optimisé grâce au lazy loading et au cache statique versionné.

6.3. Sécurité et bonnes pratiques

La conception du site Le Maine Blanc a intégré dès le départ des principes de sécurité et de qualité de code, conformément aux bonnes pratiques Django et aux standards du développement web professionnel.

Cette section présente les principales mesures mises en place.

6.3.1. Sécurité des données et des formulaires

- Protection CSRF (Cross-Site Request Forgery)

Tous les formulaires du site incluent le jeton CSRF fourni par Django afin d'empêcher l'envoi de requêtes malveillantes.

```
<form method="post" action='{% url "reservation_request" %}' novalidate>
  {% csrf_token %}
  <div class="row">...
</div>

  <p class="small text-muted mt-3">...
</p>

  <div class="text-center mt-4">...
</div>
</form>
```

Ce mécanisme vérifie que les requêtes POST proviennent bien du site légitime et non d'une source externe.

- Validation et nettoyage des données

Les formulaires, notamment le BookingForm, effectuent une validation stricte des champs (types, formats, plages de dates).

```
if start_date < today:
    raise forms.ValidationError(_("La date d'arrivée ne peut pas être antérieure à aujourd'hui."))
```

Cette étape empêche les saisies incohérentes ou malveillantes d'être enregistrées dans la base de données.

- Filtrage des champs autorisés

Avant toute sauvegarde, les vues vérifient que seules les données attendues sont traitées.

```
model_fields = [f.name for f in Booking._meta.get_fields()]
booking_data = {k: v for k, v in booking_data.items() if k in model_fields}
```

Cela protège contre les attaques de type mass assignment (injection de champs non autorisés).

6.3.2. Sécurisation des communications et de l'environnement

- HTTPS

Le site est configuré pour fonctionner sous HTTPS avec certificat SSL, assurant le chiffrement des échanges entre le serveur et les utilisateurs (formulaires, e-mails, cookies).

- Gestion sécurisée des variables sensibles

Les paramètres sensibles (mots de passe, clés secrètes, identifiants SMTP, ...) sont stockés dans un fichier .env non versionné :

```
EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD', default='')
STRIPE_SECRET_KEY = config('STRIPE_SECRET_KEY', default='')
STRIPE_PUBLIC_KEY = config('STRIPE_PUBLIC_KEY', default='')
```

Cela évite toute fuite accidentelle sur un dépôt public.

- Protection des sessions

Les données temporaires de réservation (booking_data) sont stockées dans la session Django, isolée et sécurisée par des cookies signés.

Elles sont supprimées automatiquement après confirmation.

```
if 'booking_data' in request.session:
    del request.session['booking_data']
```

6.3.3. Envoi d'e-mails sécurisé

Les messages envoyés aux clients et à l'administrateur utilisent le module EmailMessage de django.

Chaque envoi est encapsulé dans un bloc try/except pour intercepter les erreurs.

```
try:
    with translation.override('fr'):
        extra_info_1 = ""
        extra_info_2 = ""

        if booking.is_tent:
            extra_info_1 = _("Dimensions tente : {length} m x {width} m").format(...)
        elif booking.is_vehicle: ...
        if booking.electricity == 'yes':
            extra_info_2 = _("Longueur câble : {length} m").format(...)

        admin_subject = _("Nouvelle réservation de {booking.first_name} {booking.last_name}").format(booking=booking)
        admin_message_final = render_to_string('emails/admin_booking.html', { ...

        if is_render: ...
        else:
            email_admin = EmailMessage(...)
            email_admin.content_subtype = "html"
            email_admin.send(fail_silently=False)

        # Email to client in selected language
        with translation.override(request.LANGUAGE_CODE): ...

except Exception as e:
    print("⚠ Erreur lors de l'envoi des emails :", e)
    messages.warning(request, _("Votre réservation est enregistrée, mais une erreur est survenue lors de l'envoi des emails. " \
    "Veuillez contacter l'administrateur."))
```

Les e-mails sont également rendus à partir de templates HTML sécurisés (render_to_string), évitant toute injection de contenu dynamique non contrôlé.

6.3.4. Bonnes pratiques de développement

- Respect de la PEP 8 pour la lisibilité et la cohérence du code Python.
- Découpage en fonctions claires et documentées avec docstrings

```
def booking_confirm(request):
    """
    Final step:
    - Verify data integrity
    - Save booking in DB
    - Send confirmation emails (admin + customer)
    - Clear session

    Security:
    - Validate required fields before saving.
    - Remove all booking data from session after confirmation.
    """
```

- Messages utilisateurs sécurisés via le framework Django Messages (messages.success, message.error).
- Traduction et internationalisation avec les balises {% trans %} et gettext_lazy pour éviter le texte codé en dur.

- Lazy loading pour les images et minimisation des ressources statiques afin d'optimiser les performances.
- SEO et accessibilité respectés (attributs alt, balises aria-label, structure sémantique).

6.3.5. Sauvegarde et intégrité de la base de données

La base SQLite est utilisée en développement, mais la structure respecte les bonnes pratiques Django ORM :

- Intégrité référentielle gérée automatiquement (relations ForeignKey)
- Sauvegarde régulière via export dumpdata
- Nettoyage automatique des sessions expirées et des fichiers temporaires.

6.4. Points forts techniques

L'ensemble des mesures de sécurité mises en œuvre permet de garantir :

- La confidentialité des données clients
- L'intégrité des réservations
- La fiabilité des échanges (e-mails, formulaires, sessions).

Le site répond ainsi aux standards modernes de sécurité web et de développement responsable sous Django.

7. Fonctionnalités principales

Cette section présente les tests fonctionnels réalisés pour vérifier le bon fonctionnement des principales fonctionnalités du site de réservation du Camping Le Maine Blanc.

L'objectif est de démontrer que les fonctionnalités développées répondent bien aux besoins exprimés dans le cahier des charges, tout en garantissant une expérience utilisateur fluide et fiable.

7.1. Objectif du jeu d'essai

Le jeu d'essai a pour but de :

- Valider la cohérence du processus de réservation complet
- Tester la fiabilité du calcul des prix et des acomptes
- Vérifier la bonne gestion des données utilisateurs
- Confirmer la sécurité et la suppression des données temporaires après réservation
- S'assurer du bon affichage responsive sur différents supports (ordinateur, tablette, mobile).

7.2. Scénarios de tests principaux

Les tests ont été effectués manuellement via le navigateur ainsi qu'en simulation sur le serveur de développement (localhost :8000).

N°	Fonctionnalité testée	Description du test	Résultat attendu	Résultat obtenu
1	Accès au site	Saisir l'URL d'accueil	La page d'accueil s'affiche avec le menu et le logo	Conforme
2	Navigation multilingue	Changer la langue (FR/EN/ES/DE/NL)	Le contenu et les menus s'affichent dans la langue choisie	Conforme

3	Formulaire de réservation	Remplir et valider le formulaire avec des dates valides	Passage à l'étape suivante avec récapitulatif de la réservation	Conforme
4	Validation des champs obligatoires	Laisser un champ vide (ex : e-mail)	Message d'erreur « Ce champ est obligatoire »	Conforme
5	Vérification des dates	Entrer une date de fin antérieure à la date d'arrivée	Message d'erreur bloquant la soumission	Conforme
6	Calcul du prix	Réservation de 2 adultes, 1 enfant + électricité	Le prix total correspond au tarif du modèle Price et Supplement	Conforme
7	Calcul de l'acompte	Vérifier le calcul automatique de 15 % du total	L'acompte affiché = Total \times 0,15	Conforme
8	Enregistrement de la réservation	Soumettre la réservation complète	Les données sont enregistrées en base (Booking)	Conforme
9	Envoi d'e-mail client	Vérifier la réception du mail de confirmation	E-mail HTML reçu avec récapitulatif et montant	Conforme
10	Envoi d'e-mail admin	Vérifier la réception de l'e-mail de notification	E-mail HTML reçu à l'adresse admin	Conforme
11	Suppression des données temporaires	Après validation, vérifier la session	Les données booking_data sont supprimées	Conforme
12	Affichage responsive	Tester sur mobile et tablette	Le menu devient burger, mise en page fluide	Conforme

7.3. Résumé des tests

L'ensemble des tests fonctionnels réalisés sur le site se sont révélés conforme aux attentes.

Le système de réservation en ligne fonctionne correctement de bout en bout, du choix des dates à la confirmation par e-mail, sans perte de données ni erreur bloquante.

Les cas d'erreur (dates invalides, champs vides, mail incorrect) sont bien gérés et signalés à l'utilisateur via des messages clairs.

L'affichage s'adapte parfaitement aux tablettes et mobiles confirmant la réussite de l'intégration responsive avec Bootstrap 5.

7.4. Améliorations et évolutions techniques

Le site de réservation du Camping Le Maine Blanc a fait l'objet d'une couverture de tests complète, garantissant la fiabilité, la maintenabilité et la qualité du code.

Les tests unitaires et fonctionnels ont été réalisés avec pytest, en utilisant des fixtures Django et le mocking (unittest.mock.patch) pour isoler les comportements critiques (calculs, vérifications de capacité, envoi d'e-mails, intégration Stripe, ...).

Les tests couvrent :

- Les formulaires (BookingFormClassic, BookingDetailsForm, ReservationRequestForm) avec vérification des champs obligatoires, validation des données, nettoyage et normalisation des champs (emails, noms, téléphone) et les cas spécifiques comme la longueur du câble pour l'électricité ou la longueur du véhicule sont couverts
- Les modèles (Booking, Price, SupplementPrice, Capacity, MobilHome, SupplementMobilHome, SeasonInfo) en validant les règles métiers, le calcul du total et de l'acompte, vérification de la capacité pour éviter l'overbooking, création et traduction des saisons et des mobil-homes, représentation textuelle des objets,
- Les vues et URL pour vérifier les redirections, le rendu des templates, les messages utilisateurs et le bon déclenchement des processus internes (paiement, e-mails, sessions, ...),

- Les objets de contenu du site (horaires, services, tarifs) pour garantir la cohérence des données affichées et des traductions multilingues.

Les e-mails de confirmation client et d'administration sont déjà traduits automatiquement selon la langue de navigation, grâce à la gestion dynamique des fichiers .po et aux balises {% trans %} intégrées dans les templates HTML d'e-mails.

Afin de renforcer encore la qualité et la maintenabilité du projet, plusieurs évolutions sont envisagées :

- Automatiser les tests fonctionnels complets du parcours utilisateur (de la réservation à la confirmation) avec des outils comme Selenium ou Playwright pour simuler des interactions réelles sur le site,
- Générer automatiquement des récapitulatifs PDF pour le client après réservation,
- Mettre en place une surveillance d'erreurs en production (Sentry par exemple) pour détecter les comportements anormaux.

8. Veille sur les vulnérabilités de sécurité

La sécurité d'un site de réservation en ligne est un enjeu majeur pour protéger les données personnelles des utilisateurs et garantir la fiabilité des transactions. Pour le Camping Le Maine Blanc, plusieurs bonnes pratiques ont été appliquées, et une veille technique et applicative est mise en place afin de garantir la sécurité du framework Django, des dépendances Python et des bibliothèques tierces utilisées.

8.1. Sources de veille et outils utilisés

- OWASP (Open Web Application Security Project) : suivi des vulnérabilités courantes comme les injections SQL, XSS, CSRF ou les failles d'authentification,
- Django Security Releases : suivi des mises à jour de sécurité officielles de Django via le site <https://www.djangoproject.com/weblog/> pour connaître les correctifs récents et les versions corrigées,
- NVD (National Vulnerability Database) : suivi des CVE (Common Vulnerabilities and Exposures) pour les bibliothèques Python utilisées (Stripe, request),
- Pip list -outdated, python -m django -version et pip-audit : pour détecter les dépendances Python vulnérables dans le projet et les mettre à jour régulièrement.

8.2. Vulnérabilités identifiées récemment et mesures appliquées

8.2.1. Injection SQL

- Risque : un utilisateur malveillant pourrait manipuler des requêtes SQL pour accéder à des données non autorisées,
- Mesures : utilisation exclusive de l'ORM Django pour toutes les interactions avec la base de données ; aucun SQL brut n'est exécuté directement et les paramètres sont toujours passés via l'ORM ou QuerySet filtrés.

8.2.2. Cross-Site Scripting (XSS)

- Risque : injection de code JavaScript malveillant dans les formulaires ou champs de texte,
- Mesures : les templates Django utilisent automatiquement l'échappement des variables (`{{ variable }}`) et les contenus HTML dans les modèles sont validés et nettoyés si nécessaire.

8.2.3. Cross-Site Request Forgery (CSRF)

- Risque : un utilisateur malveillant pourrait forcer un utilisateur connecté à effectuer des actions non souhaitées,
- Mesures : activation du middleware `CsrfViewMiddleware` de Django et inclusion systématique des `{% csrf_token %}` dans tous les formulaires.

8.2.4. Exposition des e-mails et données personnelles

- Risque : divulgation des informations personnelles des clients lors de l'envoi d'e-mails ou stockage en base de données,
- Mesures : les e-mails sont envoyés via le serveur configuré avec TLS et les données sensibles (emails, téléphone) ne sont jamais exposées côté client et l'accès à la base de données est restreint.

8.2.5. Gestion des mot de passe et accès admin

- Risque : attaque par force brute ou vol de comptes administrateurs,
- Mesures : utilisation du système d'authentification Django avec hachage sécurisé (PBKDF2) et activation de la protection contre les tentatives de connexion répétées et recommandations de mots de passe forts.

8.3. Bonnes pratiques mises en œuvre

- Validation côté serveur et côté client pour tous les formulaires,
- HTTPS obligatoire pour toutes les pages du site afin de sécuriser les échanges,
- Suppression des données temporaires après validation d'une réservation pour éviter la fuite d'informations,
- Suivi des erreurs pour détecter toute tentative suspecte.

8.4. Plan de veille et mise à jour continue

- Mise à jour régulière de Django et des dépendances Python dès qu'un patch de sécurité est publié,
- Revue mensuelle des dépendances avec pip-audit pour détecter les vulnérabilités connues,
- Surveillance des alertes OWASP et des CVE pour les nouvelles failles affectant Django, Bootstrap, ou les bibliothèques tierces (Stripe),
- Tests réguliers des formulaires et API pour vérifier la résistance aux injections et autres attaques.

Le site est actuellement protégé contre les vulnérabilités majeures connues. La combinaison de bonnes pratiques Django, de HTTPS, de validation des formulaires et d'une veille continue permet de limiter fortement les risques pour les données clients et la plateforme de réservation.

9. Situation de recherche sur un site anglophone

9.1. **Besoin d'information**

Dans le cadre du développement du site de réservation Camping Le Maine Blanc, j'ai mené une recherche en anglais pour vérifier et renforcer les pratiques de sécurité Django appliquées au projet (configuration de settings.py, protection des formulaires, cookies, headers, gestion des clés secrètes, ...).

L'objectif était de confirmer que les mesures mises en place correspondent aux recommandations professionnelles et corriger et / ou renforcer ce qui est nécessaire.

9.2. **Méthode de recherche**

- Mots-clés utilisés : Django security best practices, Django security checklist, Django security cheat sheet, Django headers security, Django CSRF best practices
- Sites consultés :
 - <https://realpython.com/tutorials/django/>
 - <https://docs.djangoproject.com/en/5.2/howto/deployment/>
 - https://cheatsheetseries.owasp.org/cheatsheets/Django_Security_Cheat_Sheet.html
 - <https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu>
 - <https://www.deanthomson.com/blog/deploying-django-applications-with-nginx-gunicorn/>
- Critères de sélection du site final : autorité (OWASP / documentation officielle), précision technique (exemples concrets de configuration) applicable au contexte Django, clarté et exhaustivité

J'ai retenu la OWASP Django Security Cheat Sheet parce qu'elle regroupe des recommandations pratiques, point par point (headers, cookies, CSRF, XSS,

check--deploy, gestion du SECRET_KEY, ...) et fournit des exemples concrets directement exploitables dans un projet Django.

9.3. Synthèse des informations retenues (appliquées au projet)

Voici les recommandations clés extraites :

- Tenir Django et les dépendances à jour : process de mise à jour et pip-audit/Dependabot
- Ne jamais laisser DEBUG=True en production, vérification et CI/CD
- Activer les middlewares de sécurité (SecurityMiddleware, XFrameOptionsMiddleware, CsrfMiddleware) et configurer les headers (HSTS, X-Content-Type-Options)
- Configurer les cookies sécurisés (SESSION_COOKIE_SECURE, CSRF_COOKIE_SECURE) et SECURE_SSL_REDIRECT pour forcer HTTPS
- Gérer les clés secrètes hors du dépôt (variables d'environnement)

Voici comment j'ai mis ces recommandations en application au projet Camping Le Maine Blanc :

- SECRET_KEY, clés Stripe, credentials SMTP et variables d'environnement stockés en .env
- SECURE_SLL_REDIRECT, SESSION_COOKIE_SECURE et CSRF_COOKIE_SECURE activés sur l'environnement de production
- SecurityMiddleware et XFrameOptionsMiddleware inclus dans MIDDLEWARE
- Utilisation régulière de ./manage.py check --deploy pour détecter les avertissements de déploiements.

Ces actions reprennent exactement les recommandations du cheat-sheet OWASP.

10. Extrait du site anglophone et traduction

10.1. Extrait (anglais) : source OWASP Django Security Cheat Sheet

« Introduction

The Django framework is a powerful Python web framework, and it comes with built-in security features that can be used out-of-the-box to prevent common web vulnerabilities. This cheat sheet lists actions and security tips developers can take to develop secure Django applications. It aims to cover common vulnerabilities to increase the security posture of your Django application. Each item has a brief explanation and relevant code samples that are specific to the Django environment.

The Django framework provides some built-in security features that aim to be secure-by-default. These features are also flexible to empower a developer to re-use components for complex use-cases. This opens up scenarios where developers unfamiliar with the inner workings of the components can configure them in an insecure way. This cheat sheet aims to enumerate some such use cases.

General Recommendations

- *Always keep Django and your application's dependencies up-to-date to keep up with security vulnerabilities.*
- *Ensure that the application is never in DEBUG mode in a production environment. Never run `DEBUG = True` in production.*
- *Use packages like [django_ratelimit](#) or [django-axes](#) to prevent brute-force attacks.*

Authentication

- *Use `django.contrib.auth` app for views and forms for user authentication operations such as login, logout, password change, etc. Include the module*

and its

dependencies `django.contrib.contenttypes` and `django.contrib.sessions` in the `INSTALLED_APPS` setting in the `settings.py` file.

- Use the `@login_required` decorator to ensure that only authenticated users can access a view. The sample code below illustrates usage of `@login_required`.*
- Use password validators for enforcing password policies. Add or update the `AUTH_PASSWORD_VALIDATORS` setting in the `settings.py` file to include specific validators required by your application.*
- Store passwords using `make_password` utility function to hash a plain-text password.*
- Check a plaintext password against a hashed password by using the `check_password` utility function. »*

10.2. Traduction de l'extrait en français

« Introduction

Le framework Django est un puissant framework Python, et il comprend des fonctions de sécurité intégrées qui peuvent être utilisées pour prévenir des vulnérabilités web courantes. Cette fiche aide-mémoire liste les actions et les conseils de sécurité à faire pour développer une application Django sécurisée. Il vise à couvrir les vulnérabilités communes afin d'augmenter la posture de sécurité de votre application Django. Chaque article contient une brève explication et un exemple de code qui est spécifique à l'environnement Django.

Le framework Django fournit plusieurs fonctions de sécurité par défaut. Ces fonctions sont aussi flexibles pour permettre aux développeurs de réutiliser des composants pour des utilisations plus complexes. Cette ouverture de scénarios où des développeurs qui ne sont pas familiarisés avec le fonctionnement interne des composants peuvent les configurer de manière non sécurisée. Cet aide-mémoire vise à énumérer certains de ces cas d'utilisation.

Recommandations générales

- Toujours garder Django et les dépendances de votre application à jour pour rester informer des vulnérabilités de sécurité.*

- *S'assurer que l'application n'est jamais en mode DEBUG dans l'environnement de production. Ne jamais lancer `DEBUG=True` en production.*
- *Utiliser des packages comme `django_ratelimit` ou `django-axes` pour prévenir des attaques par force brute.*

Authentification

- *Utilise l'application `django.contrib.auth` pour les vues et les formulaires pour les opérations d'authentification d'utilisateur comme la connexion, la déconnexion, le changement de mot de passe, ... Inclure le module et ses dépendances `django.contrib.contenttypes` et `django.contrib.sessions` dans les paramètres `INSTALLED_APPS` du fichier `settings.py`.*
- *Utilise le décorateur `@login_required` pour s'assurer que seulement les utilisateurs authentifiés peuvent accéder à une vue. L'exemple de code ci-dessous illustre l'utilisation de `@login_required`.*
- *Utilise les validateurs de mot de passe pour appliquer les politiques de mot de passe. Ajoute ou met à jour le paramètre `AUTH_PASSWORD_VALIDATORS` dans le fichier `settings.py` pour inclure les validations spécifiques requises par votre application.*
- *Stocke les mots de passe en utilisant la fonction `make-password` pour hacher le mot de passe brut.*
- *Vérifie un mot de passe en brut contre un mot de passe hacher en utilisant la fonction `check-password`. »*

11. Conclusion

Le développement de site web du Camping Le Maine Blanc a constitué une expérience complète et formatrice, couvrant l'ensemble du cycle de conception d'une application web moderne : de la définition des besoins à la mise en ligne, en passant par le développement, la sécurisation, les tests et l'optimisation.

Ce projet a permis de mettre en pratique l'ensemble des compétences acquises au cours de la formation de Développeur Web et Web Mobile, notamment :

- La conception d'une architecture Django robuste et modulaire,
- L'intégration d'un front-end responsive avec Bootstrap 5,
- La mise en place d'un système de réservation dynamique,
- La gestion de la sécurité des données et des paiements,
- La création d'une interface multilingue et optimisée pour le référencement (SEO).

Une attention particulière a été portée à la qualité du code, à la sécurité (protection CSRF, validation des formulaires, gestion des clés et des sessions, configuration HTTPS, ...) et à fonctionnels complets.

Ce projet m'a également permis de développer des compétences transversales essentielles au métier de développeur :

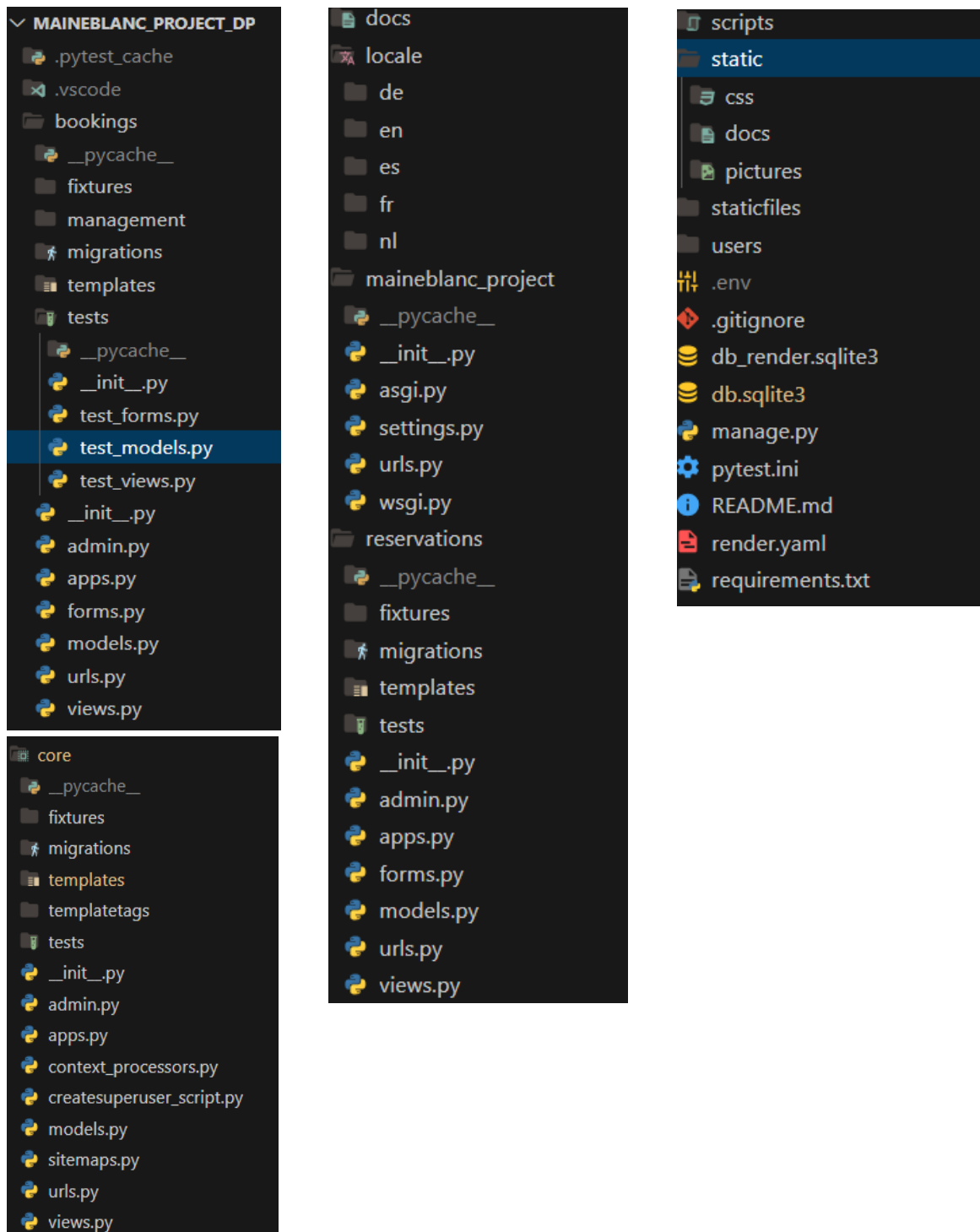
- Autonomie dans la recherche de solutions techniques,
- Utilisation de la documentation officielle et de ressources anglophones (Real Python, OWASP),
- Rigueur dans l'écriture et la maintenance du code, capacité à concevoir une application professionnelle répondant à des besoins clients réels.

En conclusion, la réalisation de ce site web professionnel représente l'aboutissement de mon parcours de formation et la validation concrète de mes compétences techniques et méthodologiques dans le développement web.

Ce projet marque une étape importante dans mon évolution professionnelle et me prépare à exercer pleinement le métier de développeur web et mobile.

12. Annexes

12.1. Arborescence du projet Django

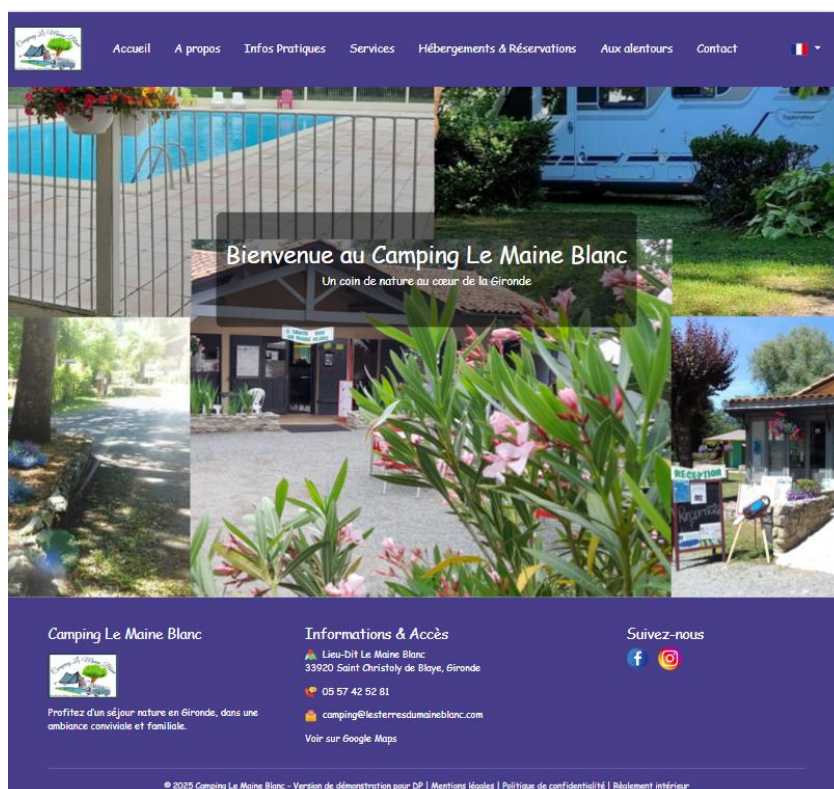


12.2. Technologies utilisées



Type	Outil / Technologie	Utilisation principale
Langage	Python 3 / JavaScript	Back-end (Django) / Interactivité
Framework	Django 5 / Bootstrap 5	Structure web et responsive design
Base de données	SQLite	Environnement de développement
Serveur de test	Django runserver	Exécution locale
Versioning	Git / GitHub	Suivi des versions et sauvegarde du projet
Tests	Pytest / Django TestCase	Vérification du fonctionnement des modèles
Traduction	Django Parler / gettext	Gestion multilingue (FR, EN, ES, DE, NL)
Outils de développement	VS Code	IDE principal
Déploiement (prévu)	Gunicorn + Nginx sauf si choix autre des gérants pour l'hébergeur	Hébergement futur en production

12.3. Exemples d'interfaces

Page d'accueil :



Formulaire de réservation :

[Accueil](#) [A propos](#) [Infos Pratiques](#) [Services](#) [Hébergements & Réservations](#) [Aux alentours](#) [Contact](#) 

Réserver un emplacement

Pour les réservations ouvriers, merci de contacter directement le camping au 05 57 42 52 81 ou via notre formulaire de contact.

Emplacements *
Type d'emplacement
--- Choisissez ---


Dates du séjour *
Date d'arrivée :
Date de départ :



Nombre d'occupants *
Adultes : Enfants +8 ans : Enfants -8 ans : Animaux (2 max) :



Electricité *
☒ Avec électricité ☐ Sans électricité

Réserver

* Champ obligatoires

**Camping Le Maine Blanc**

Informations & Accès
 Lieu-Dit Le Maine Blanc
33920 Saint Christoly de Blaye, Gironde
 05 57 42 52 81

Suivez-nous
 

Récapitulatif de réservation :

Réservation confirmée

Dates :
31/10/2025 — 01/11/2025

VOS COORDONNÉES
Nom : Doe
Prénom : John
Adresse mail : johndoe@test.com
Téléphone : 0606060606

NOMBRE DE PERSONNES
Adultes : 2
Enfants + 8 ans : 0
Enfants - 8 ans : 1
Animaux : 1

EMPLACEMENT / VÉHICULE
Type d'emplacement : Caravane
Longueur du véhicule : 6,00 m
Électricité : Avec électricité
Câble électrique : 2,5 m

Total à payer : 24,40 €
Acompte versé : 3,66 €
Solde restant à payer le jour de l'arrivée : 20,74 €

Faire une nouvelle réservation

Interface d'administration Django :

The screenshot shows the Django administration interface. At the top, a blue header bar contains the title 'Administration de Django' and a navigation menu with links: 'BIENVENUE, DEMO_ADMIN_DP', 'VOIR LE SITE', 'MODIFIER LE MOT DE PASSE', and 'DÉCONNEXION'. Below the header, the page is titled 'Site d'administration'. The main content area is divided into two columns. The left column contains three sections: 'AUTHENTIFICATION ET AUTORISATION' with links for 'Groupes' and 'Utilisateurs'; 'INFORMATIONS DIVERSES' with links for 'Laverie', 'Modalités du camping', 'Piscine', and 'Restauration'; and 'PRIX ET RÉSERVATIONS' with links for 'Capacités d'emplacements', 'Dates des saisons', 'Mobil-homes', 'Prix des Suppléments', 'Réservations', 'Suppléments mobil-home', 'Tarifs', and 'Tarifs et infos divers'. Each link is accompanied by '+ Ajouter' and 'Modification' icons. The right column contains a sidebar with 'Actions récentes' and 'Mes actions' (Aucun(e) disponible).

12.4. Liens utiles

- Documentation officielle Django : <https://docs.djangoproject.com/>
- Documentation Bootstrap 5 : <https://getbootstrap.com/docs/5.3/>
- OWASP Cheat Sheet - Django Security : <https://cheatsheetseries.owasp.org/>
- GitHub du projet : https://github.com/Nathi33/maineblanc_project_DP.git
- Maquettage complet : https://github.com/Nathi33/maineblanc_project/releases/tag/v1.0-maquettage
- Version de démonstration déployée sur Render : <https://maineblanc-project-dp.onrender.com>
- Administration du site :
 - <https://maineblanc-project-dp.onrender.com/admin/>
 - Identifiants de connexion (démonstration) :
 - Nom d'utilisateur : Demo_admin_DP
 - Adresse mail : admin_dp@test.com
 - Mot de passe : Demo1234 !

12.5. Glossaire

- Back-end : partie serveur d'un site web gérant la logique et la base de données
- Front-end : partie visible du site (interface utilisateur)
- Responsive Design : Adaptation automatique de l'affichage selon la taille de l'écran
- Framework : ensemble d'outils facilitant le développement d'applications
- ORM (Object Relational Mapper) : outil permettant d'interagir avec la base de données via des objets Python
- Test unitaire : test automatisé qui vérifie le bon fonctionnement d'une partie du code
- Session : données temporaires stockées côté serveur pendant la navigation