

DOSSIER DE PROJET

Nom : MONLOUIS-FÉLICITÉ

Prénom : Yoann

Promotion : Novembre 2024 CEF

**Titre Professionnel Développeur Web et Web Mobile
(DWWM)**

**Projet : Application de gestion de rendez-vous pour un
salon de tatouage**

Sommaire

1. Présentation du projet	3
1.1 Contexte	3
1.2 Objectifs du projet	3
1.3 Public cible	3
2. Analyse des besoins	4
2.1 Besoins fonctionnels	4
2.2 Contraintes métier	4
3. Maquettage des interfaces utilisateur (Figma)	5
4. Réalisation des interfaces utilisateur	7
4.1 Interfaces statiques	7
4.2 Interfaces dynamiques	7
5. Modélisation de la base de données	8
5.1 Modèle Conceptuel de Données (MCD)	8
5.2 Entité du MCD	8
5.3 Règles de gestion exprimées par le MCD	9
5.4 Modèle Logique de Données (MLD)	10
6. Mise en place de la base de données relationnelle	12
7. Développement des composants d'accès aux données	13
8. Logique métier – Calcul des disponibilités	14
9. Gestion des rendez-vous (CRUD)	16
9.1 Parcours côté client : création et suivi	16
9.2 Parcours côté administrateur : validation et gestion complète (CRUD)	17
9.3 Sécurisation des opérations CRUD	18
10. Sécurité de l'application	19
11. Tests et validation	21
12. Conclusion et évolutions possibles	22

1. Présentation du projet

1.1 Contexte

Dans le cadre de la préparation à l'épreuve du Titre Professionnel Développeur Web et Web Mobile, j'ai réalisé une application web permettant la gestion des rendez-vous pour un salon de tatouage fictif nommé Need Ink.

Les salons de tatouage fonctionnent généralement sur rendez-vous, avec des contraintes spécifiques : durée variable des prestations, disponibilité des artistes, validation des rendez-vous par l'administrateur. L'objectif du projet est de proposer une solution simple et sécurisée pour gérer ces contraintes.

1.2 Objectifs du projet

L'objectif de ce projet est de concevoir une application web permettant aux clients d'un salon de tatouage de créer un compte et de demander un rendez-vous en ligne. L'application offre la possibilité de visualiser les créneaux disponibles en fonction de l'artiste choisi, de la prestation sélectionnée et de la date souhaitée.

Elle permet également à un administrateur de gérer l'ensemble des rendez-vous en confirmant, modifiant ou annulant les demandes. L'un des objectifs majeurs du projet est de garantir la cohérence du planning du salon, en empêchant tout chevauchement de rendez-vous ou réservation sur un créneau indisponible.

1.3 Public cible

L'application s'adresse principalement à deux types d'utilisateurs. Les clients du salon utilisent l'application pour consulter les disponibilités des artistes et demander un rendez-vous en ligne. L'administrateur du salon utilise quant à lui l'interface pour gérer le planning, valider les demandes et assurer une organisation optimale des rendez-vous.

2. Analyse des besoins

2.1 Besoins fonctionnels

L'application doit permettre une authentification sécurisée des utilisateurs afin de protéger l'accès aux fonctionnalités. Elle doit distinguer les rôles CLIENT et ADMIN afin d'adapter les interfaces et les droits d'accès. Les utilisateurs doivent pouvoir consulter la liste des artistes ainsi que les prestations proposées. Le système doit calculer dynamiquement les disponibilités en fonction de plusieurs critères et permettre la création de rendez-vous associés à un statut. Enfin, l'administrateur doit disposer d'outils lui permettant de gérer les rendez-vous selon les principes du CRUD.

2.2 Contraintes métier

Chaque prestation possède une durée fixe définie en base de données. Un artiste ne peut assurer qu'un seul rendez-vous à la fois, ce qui implique un contrôle strict des chevauchements. Les rendez-vous doivent également respecter les périodes d'indisponibilité définies pour chaque artiste. Lors de sa création, un rendez-vous est automatiquement enregistré avec le statut *PENDING*. Seul un administrateur est autorisé à confirmer, modifier ou annuler un rendez-vous.

3. Maquettage des interfaces utilisateur (Figma)

Des maquettes ont été réalisées à l'aide de l'outil Figma afin de définir la structure des pages et le parcours utilisateur avant le développement. Ce travail de maquettage a permis d'anticiper l'organisation des contenus et d'assurer une navigation claire et intuitive. Quatre pages principales ont été maquetées : la page de connexion, la page de réservation, la page d'accueil client et la page d'administration.

Maquette de la page de connexion. Le header est noir avec 'Need Ink' à gauche et 'Connexion' à droite. Le titre principal est 'Connexion'. Il y a deux champs de saisie : 'Email' et 'Mot de passe'. En dessous, un bouton 'Se connecter'.

Maquette Figma – Page Connexion

Maquette de la page de réservation. Le header est noir avec 'Need Ink' à gauche, 'Nom de l'utilisateur connecté' au centre et 'Deconnexion' à droite. Le titre principal est 'Réserver'. Il y a trois champs de saisie : 'Artiste', 'Prestation' et 'Date'. En dessous, un bouton 'Afficher les disponibilités'. Le titre secondaire est 'Disponibilités'. Il y a un champ de sélection 'Choisir un Créneau' avec une valeur '10:00 → 11:00' et un bouton 'Confirmer'.

Maquette Figma – Page Réservation

Need Ink

Nom de l'utilisateur connecté

Deconnexion

Accueil

Bienvenue Utilisateur

Reserver un RDV

Vos RDV à venir

Date	Prestation	Artiste	Statut

Maquette Figma – Page Accueil Client

Need Ink

Nom de l'utilisateur connecté

Deconnexion

Gestion des RDV

Date	Client	Prestation	Artiste	Statut	Action
13/10/2024	Gino Pino	Consultation	Daniela D	En attente	Confirmer
					Annuler
					Modifier
17/10/2024	Marvin Martin	Session 2H	Sasha P	En attente	Confirmer
					Annuler
					Modifier

Maquette Figma – Page Admin

4. Réalisation des interfaces utilisateur

4.1 Interfaces statiques

Les interfaces utilisateur de l'application ont été conçues en HTML et CSS afin de proposer une structure claire, lisible et cohérente sur l'ensemble des pages. Une attention particulière a été portée à l'organisation des contenus afin de faciliter la compréhension et la navigation pour l'utilisateur. Les pages reposent sur une structure sémantique simple, permettant d'identifier rapidement les différentes zones fonctionnelles telles que l'en-tête, le contenu principal et les actions disponibles.

Les formulaires ont été conçus de manière logique et progressive, en regroupant les champs par fonction. Cette organisation permet de limiter les erreurs de saisie et d'améliorer l'expérience utilisateur lors de la connexion ou de la réservation d'un rendez-vous. Les libellés sont explicites et les champs obligatoires clairement identifiés.

Les données sont principalement présentées sous forme de tableaux, notamment dans les interfaces de consultation et d'administration. Ces tableaux facilitent la lecture des informations essentielles telles que les dates, les artistes, les prestations et les statuts des rendez-vous. Leur structure permet une visualisation rapide et efficace des données, ce qui est particulièrement utile pour la gestion administrative.

Les menus de navigation ont été conçus pour s'adapter au rôle de l'utilisateur connecté. Les liens affichés varient selon qu'il s'agisse d'un client ou d'un administrateur, ce qui permet de simplifier l'interface et d'éviter l'accès à des fonctionnalités non autorisées.

4.2 Interfaces dynamiques

La partie dynamique des interfaces est assurée par PHP, qui permet de générer les contenus en fonction du contexte utilisateur et des données issues de la base de données. L'affichage des pages est conditionné par l'état de connexion et par le rôle de l'utilisateur, garantissant ainsi une interface adaptée à chaque profil.

Les formulaires interagissent directement avec la base de données via des traitements côté serveur. Lorsqu'une action est effectuée par l'utilisateur, comme la création d'un rendez-vous ou la mise à jour d'un statut, les données sont traitées, validées et enregistrées avant que l'interface ne soit mise à jour. Cette interaction permet de refléter immédiatement les changements effectués, offrant une expérience utilisateur fluide et cohérente.

Des messages de confirmation ou d'erreur sont affichés à l'utilisateur à la suite de chaque action importante. Ces messages permettent d'informer clairement l'utilisateur du résultat de son action, qu'il s'agisse d'une réussite ou d'un problème rencontré, et participent à la compréhension du fonctionnement de l'application.

Enfin, certaines interfaces reposent sur une mise à jour dynamique des données, notamment lors de la consultation des disponibilités. Les contenus affichés évoluent en fonction des choix de l'utilisateur (artiste, prestation, date), ce qui permet de proposer une interaction dynamique tout en conservant une logique de traitement côté serveur.

5. Modélisation de la base de données

5.1 Modèle Conceptuel de Données (MCD)

Le modèle conceptuel de données repose sur plusieurs entités principales : les utilisateurs, les artistes, les prestations, les rendez-vous et les indisponibilités. Les relations entre ces entités traduisent les règles métier, notamment le fait qu'un utilisateur peut avoir plusieurs rendez-vous, qu'un artiste peut réaliser plusieurs prestations et qu'un rendez-vous est associé à une prestation précise.

5.2 Entité du MCD

USER_ACCOUNT

- id_user (identifiant)
- email
- firstname
- lastname
- telephone
- role
- created_at

Représente un client ou un administrateur du salon.

ARTIST

- id_artist (identifiant)
- artist_name
- bio
- speciality
- is_active

Représente un tatoueur travaillant dans le salon.

SERVICE

- id_service (identifiant)
- service_name
- duration_min
- is_active

Représente une prestation proposée par le salon, avec une durée fixe.

APPOINTMENT

- id_appointment (identifiant)
- start_at
- end_at
- status
- created_at

Représente une demande ou une réservation effective.

UNAVAILABILITY

- id_unavailability (identifiant)
- start_at
- end_at
- reason

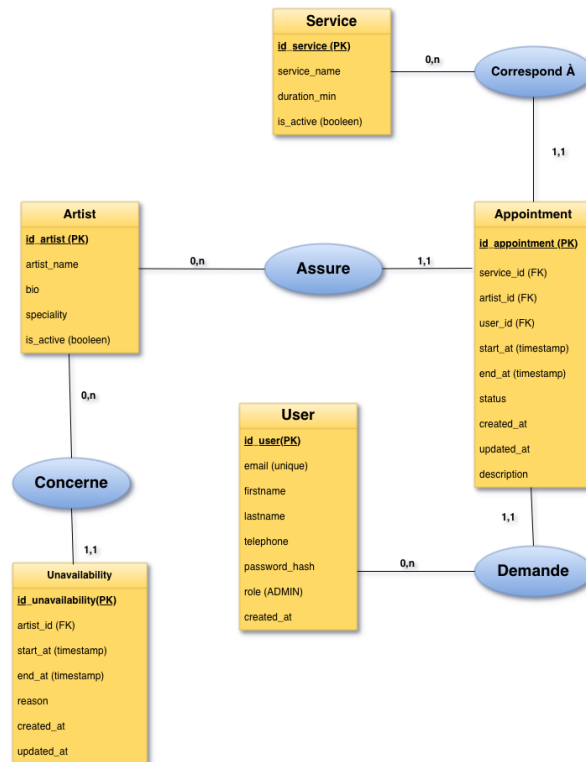
Représente une période durant laquelle un artiste n'est pas disponible.

5.3 Règles de gestion exprimées par le MCD

Le MCD met en évidence plusieurs règles métier importantes. Un rendez-vous est obligatoirement lié à un utilisateur, un artiste et une prestation. Un artiste ne peut pas être réservé sur un créneau correspondant à une période d'indisponibilité. La durée d'un rendez-vous dépend exclusivement de la prestation choisie. Enfin, un utilisateur peut effectuer plusieurs demandes de rendez-vous sans limitation.

5.4 Modèle Logique de Données (MLD)

Le modèle logique de données traduit le MCD en tables relationnelles. Chaque table est définie avec une clé primaire, des clés étrangères et des contraintes d'intégrité permettant de garantir la cohérence des données stockées en base.



ARTIST (0,n) assure (1,1) APPOINTMENT

Un artiste assure des rendez-vous.

Chaque rendez-vous est obligatoirement assuré par un seul artiste, tandis qu'un artiste peut assurer zéro, un ou plusieurs rendez-vous.

APPOINTMENT (1,1) correspond à (0,n) SERVICE

Un rendez-vous correspond à un service.

Chaque rendez-vous est associé à un unique service, tandis qu'un service peut donner lieu à zéro, un ou plusieurs rendez-vous.

UNAVAILABILITY (1,1) concerne (0,n) ARTIST

Une indisponibilité concerne un artiste.

Chaque indisponibilité est rattachée à un seul artiste, tandis qu'un artiste peut être concerné par zéro, une ou plusieurs périodes d'indisponibilité.

USER (client) (0,n) Demande (1,1) APPOINTMENT

Un utilisateur (client) demande des rendez-vous.

Un rendez-vous est demandé par un seul utilisateur, tandis qu'un utilisateur peut demander zéro, un ou plusieurs rendez-vous.

6. Mise en place de la base de données relationnelle

La base de données de l'application a été conçue et implémentée en MySQL afin de stocker de manière structurée l'ensemble des données nécessaires au fonctionnement du système de réservation. Elle repose sur un modèle relationnel permettant de garantir la cohérence, la fiabilité et la pérennité des données.

Des contraintes métier ont été définies directement au niveau de la base de données afin de prévenir les incohérences et les erreurs de saisie. La durée des prestations est ainsi contrôlée par une contrainte garantissant qu'elle soit strictement positive. Ce choix permet d'éviter l'enregistrement de prestations invalides et assure la cohérence des calculs de disponibilités réalisés côté serveur.

La cohérence temporelle des rendez-vous est également assurée par une contrainte imposant que la date et l'heure de fin d'un rendez-vous soient toujours postérieures à celles du début. Cette règle garantit qu'aucun rendez-vous ne puisse être enregistré avec des dates incohérentes, ce qui pourrait compromettre le calcul du planning.

Les statuts des rendez-vous sont strictement encadrés afin de refléter le cycle de vie d'une demande de réservation. Un rendez-vous peut être en attente de validation (*PENDING*), confirmé (*CONFIRMED*), refusé (*REFUSED*) ou annulé (*CANCELLED*). Le contrôle de ces valeurs permet d'éviter l'utilisation de statuts non prévus par la logique métier et facilite la gestion des rendez-vous dans l'application.

Enfin, l'intégrité référentielle est assurée par l'utilisation de clés étrangères reliant les tables entre elles. Chaque rendez-vous est associé à un utilisateur, un artiste et une prestation existants en base de données. Ces relations garantissent qu'aucune donnée orpheline ne puisse être créée et assurent la cohérence globale du système.

L'ensemble de ces contraintes permet de renforcer la fiabilité de l'application en déléguant une partie des règles métier à la base de données, en complément des contrôles réalisés côté serveur.

7. Développement des composants d'accès aux données

L'accès aux données de l'application est réalisé à l'aide de PDO (PHP Data Objects), une extension native de PHP permettant d'interagir avec une base de données relationnelle de manière sécurisée et performante. Le choix de PDO garantit une séparation claire entre la logique applicative et l'accès aux données, tout en offrant un haut niveau de sécurité.

Toutes les requêtes SQL utilisées dans l'application sont exécutées à l'aide de requêtes préparées. Cette approche permet de protéger l'application contre les injections SQL en séparant strictement le code SQL des données fournies par l'utilisateur. Les paramètres sont liés dynamiquement aux requêtes, ce qui empêche toute interprétation malveillante des entrées utilisateur.

Les composants d'accès aux données ont été conçus afin de répondre aux besoins fonctionnels de l'application. Ils permettent notamment la récupération des informations relatives aux rendez-vous, aux utilisateurs, aux artistes et aux prestations. Pour cela, des requêtes SQL incluant des jointures sont utilisées afin de regrouper les données issues de plusieurs tables en une seule requête cohérente.

Cette utilisation des jointures améliore les performances de l'application en limitant le nombre d'appels à la base de données et en centralisant la récupération des informations nécessaires à l'affichage des interfaces. Elle garantit également la cohérence des données affichées, en s'appuyant sur les relations définies entre les tables.

Enfin, l'accès aux données est centralisé à travers une fonction dédiée à la connexion à la base de données. Cette approche facilite la maintenance de l'application, permet une évolution plus simple du schéma de données et renforce la lisibilité du code. L'ensemble des composants d'accès aux données a été conçu dans une logique de robustesse, de sécurité et de clarté, conformément aux bonnes pratiques du développement web.

```
1  <?php
2  // config/db.php
3  declare(strict_types=1);
4
5  function db(): PDO {
6      static $pdo = null;
7      if ($pdo instanceof PDO) return $pdo;
8
9      $pdo = new PDO(
10         "mysql:host=127.0.0.1;port=8889;dbname=NeedInk;charset=utf8mb4",
11         "root",
12         "root",
13         [
14             PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
15             PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
16             PDO::ATTR_EMULATE_PREPARES => false,
17         ]
18     );
19
20     return $pdo;
21 }
```

Connexion centralisée à la base de données via PDO, utilisée par l'ensemble de l'application.

8. Logique métier – Calcul des disponibilités

La logique de calcul des disponibilités constitue le cœur fonctionnel de l'application. Elle répond à un besoin métier essentiel du salon de tatouage : proposer aux clients uniquement des créneaux réellement disponibles, tout en garantissant la cohérence du planning des artistes.

Lorsqu'un utilisateur souhaite effectuer une réservation, il commence par sélectionner un artiste, une prestation et une date. Ces informations sont transmises au serveur afin de déclencher le calcul des disponibilités. La durée de la prestation sélectionnée est alors récupérée depuis la base de données. Cette durée, exprimée en minutes, sert d'unité de base pour la génération des créneaux horaires.

```
23 // --- Paramètres GET (sélection) ---
24 $id_artist = isset($_GET['id_artist']) ? (int)$_GET['id_artist'] : 0;
25 $id_service = isset($_GET['id_service']) ? (int)$_GET['id_service'] : 0;
26 $date      = $_GET['date'] ?? (new DateTimeImmutable('today'))->format('Y-m-d');
```

Récupération des paramètres nécessaires au calcul des disponibilités (artiste, prestation, date).

L'application définit ensuite une plage horaire de travail pour la journée sélectionnée, correspondant aux horaires d'ouverture du salon. À partir de cette plage, des créneaux successifs sont générés dynamiquement en ajoutant la durée de la prestation à chaque point de départ possible. Chaque créneau est ainsi caractérisé par une date et une heure de début (`start_at`) et une date et une heure de fin (`end_at`).

```
$cursor = $open;
$step = new DateInterval('PT' . $durationMin . 'M');

while ($cursor < $close) {
    $end = $cursor->add($step);
    if ($end > $close) break;

    $blocked = false;

    foreach ($appointments as [$aS, $aE]) {
        if (overlap($cursor, $end, $aS, $aE)) { $blocked = true; break; }
    }

    if (!$blocked) {
        foreach ($unavs as [$uS, $uE]) {
            if (overlap($cursor, $end, $uS, $uE)) { $blocked = true; break; }
        }
    }

    if (!$blocked) {
        $slots[] = ['start' => $cursor, 'end' => $end];
    }

    $cursor = $cursor->add($step);
}
```

Génération dynamique des créneaux horaires en fonction de la durée de la prestation

Chaque créneau généré est soumis à une série de contrôles visant à garantir sa validité. Dans un premier temps, l'application vérifie l'absence de conflit avec les rendez-vous existants de l'artiste. Seuls les rendez-vous ayant le statut *PENDING* ou *CONFIRMED* sont pris en compte, afin d'exclure les rendez-vous refusés ou annulés du calcul. Un créneau est considéré comme invalide dès lors qu'un chevauchement temporel est détecté avec un rendez-vous existant.

Dans un second temps, l'application contrôle les périodes d'indisponibilité définies pour l'artiste. Si un créneau chevauche une période d'indisponibilité, il est automatiquement exclu. Cette double vérification permet de garantir que les créneaux proposés respectent à la fois les contraintes du planning et les absences planifiées des artistes.

Les contrôles de chevauchement reposent sur une logique de comparaison entre les dates de début et de fin des créneaux, des rendez-vous et des indisponibilités. Cette logique permet d'identifier précisément les intersections temporelles et d'éliminer les créneaux incompatibles.

```
foreach ($appointments as [$aS, $aE]) {  
    if (overlap($cursor, $end, $aS, $aE)) { $blocked = true; break; }  
}  
  
if (!$blocked) {  
    foreach ($unavs as [$uS, $uE]) {  
        if (overlap($cursor, $end, $uS, $uE)) { $blocked = true; break; }  
    }  
}
```

Détection des chevauchements entre les créneaux générés, les rendez-vous existants et les périodes d'indisponibilité.

À l'issue de ces traitements, seuls les créneaux valides sont conservés et affichés à l'utilisateur sous forme de liste déroulante. Cette approche empêche toute réservation sur un créneau déjà occupé ou indisponible et garantit une expérience utilisateur fiable.

L'ensemble du calcul des disponibilités est réalisé côté serveur afin de centraliser la logique métier, d'assurer la fiabilité des données et d'éviter toute manipulation ou contournement côté client.

```
<select name="start_at" required>  
    <?php foreach ($slots as $sl): ?>  
        <?php  
            $startStr = $sl['start']->format('Y-m-d H:i:s');  
            $label = $sl['start']->format('H:i') . ' → ' . $sl['end']->format('H:i');  
        ?>  
        <option value="<?= e($startStr) ?>"><?= e($label) ?></option>  
    <?php endforeach; ?>  
</select>
```

Affichage des créneaux disponibles après application de l'ensemble des règles métier.

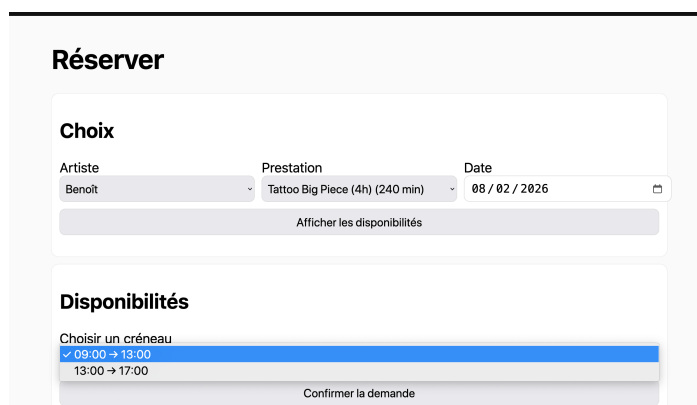
9. Gestion des rendez-vous (CRUD)

La gestion des rendez-vous repose sur un fonctionnement en deux temps afin de répondre à l'organisation réelle d'un salon de tatouage. Le client effectue une demande de rendez-vous, puis l'administrateur valide ou ajuste cette demande avant qu'elle ne devienne définitive. Ce choix permet de garder un contrôle sur le planning, tout en offrant une prise de rendez-vous en ligne simple pour l'utilisateur.

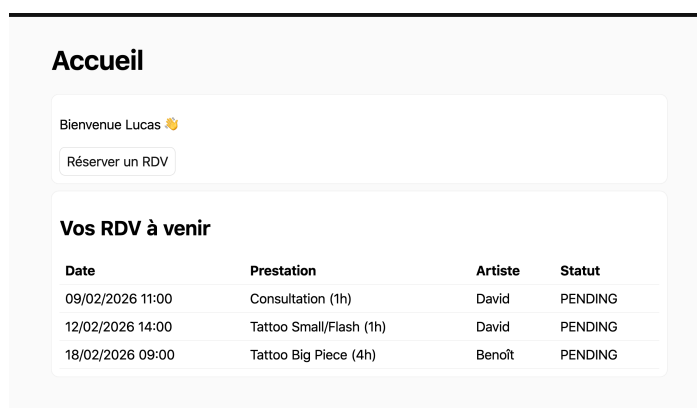
9.1 Parcours côté client : création et suivi

Du côté client, l'application permet d'abord de **créer une demande de rendez-vous** à partir de la page de réservation. Le client sélectionne un artiste, une prestation et un créneau proposé par le système de disponibilités. Lors de l'enregistrement, le rendez-vous est automatiquement créé avec le statut **PENDING**. Ce statut indique que la demande est en attente de validation par l'administrateur et qu'elle n'est pas encore confirmée.

Le client peut ensuite **consulter ses rendez-vous à venir** depuis l'interface d'accueil. Cette liste affiche les informations principales (date/heure, artiste, prestation et statut), ce qui permet à l'utilisateur de suivre l'avancement de sa demande. Le statut joue ici un rôle important, car il informe immédiatement le client si sa demande est confirmée, refusée ou annulée.



Création d'un RDV côté client (statut PENDING)



Date	Prestation	Artiste	Statut
09/02/2026 11:00	Consultation (1h)	David	PENDING
12/02/2026 14:00	Tattoo Small/Flash (1h)	David	PENDING
18/02/2026 09:00	Tattoo Big Piece (4h)	Benoît	PENDING

Consultation des rendez-vous et suivi du statut

9.2 Parcours côté administrateur : validation et gestion complète (CRUD)

Du côté administrateur, l'application fournit une interface de gestion centralisée permettant de visualiser l'ensemble des rendez-vous. Cette page regroupe les informations liées au rendez-vous grâce à des jointures (client, prestation, artiste, dates et statut). L'administrateur dispose ensuite d'actions permettant de faire évoluer le rendez-vous tout au long de son cycle de vie.

Dans ce contexte, plusieurs opérations sont possibles :

- **Confirmer** un rendez-vous correspond à une validation de la demande : le rendez-vous passe de *PENDING* à **CONFIRMED**. Cette action officialise le créneau dans le planning.
- **Refuser** un rendez-vous permet de répondre négativement à une demande, par exemple si le créneau n'est plus compatible avec l'organisation du salon. Le statut devient **REFUSED**.
- **Annuler** un rendez-vous correspond à une suppression fonctionnelle (soft delete). Le rendez-vous est conservé en base pour l'historique, mais il est considéré comme annulé via le statut **CANCELLED**.
- **Modifier** un rendez-vous permet d'ajuster la réservation. L'administrateur peut changer la date/heure, l'artiste ou la prestation. Lors d'une modification, l'application recalcule automatiquement l'heure de fin (*end_at*) en fonction de la durée de la prestation sélectionnée. Des contrôles sont réalisés afin d'empêcher toute modification entraînant un chevauchement avec un autre rendez-vous ou une indisponibilité.

Cette gestion par statuts permet de conserver un historique clair, d'éviter la suppression brutale des données et de garantir la cohérence du planning.

18/02/2026 09:00	Lucas Martin client1@needink.test	Tattoo Big Piece (4h) (240 min)	Benoît	PENDING	<div>Confirmer Refuser</div> <div>Annuler</div> <div>Modifier</div> <div>Début 18 / 02 / 2026 09 : 00</div> <div>Artiste Benoît</div> <div>Prestation Tattoo Big Piece (4h) (240 min)</div> <div>Enregistrer</div> <div>La fin (<i>end_at</i>) est recalculée automatiquement selon la durée du service.</div>
---------------------	--	---------------------------------------	--------	---------	--

Gestion administrative du cycle de vie du rendez-vous (CRUD)

9.3 Sécurisation des opérations CRUD

Toutes les actions de gestion des rendez-vous sont réalisées via des requêtes **POST**, centralisées dans un script d'actions serveur. L'accès à ces opérations est restreint aux utilisateurs disposant du rôle **ADMIN**. Les formulaires sont protégés par un token CSRF afin d'éviter qu'une action sensible soit exécutée sans l'accord réel de l'utilisateur connecté. Les requêtes SQL correspondantes utilisent des requêtes préparées afin de garantir la sécurité des traitements.

10. Sécurité de l'application

La sécurité de l'application a été prise en compte dès la conception afin de protéger les données des utilisateurs et de garantir un accès contrôlé aux différentes fonctionnalités. Plusieurs mécanismes complémentaires ont été mis en œuvre côté serveur pour répondre aux principaux risques liés aux applications web.

Les mots de passe des utilisateurs ne sont jamais stockés en clair en base de données. Lors de la création d'un compte, les mots de passe sont chiffrés à l'aide de la fonction `password_hash`, qui applique un algorithme de hachage sécurisé. Lors de l'authentification, la fonction `password_verify` est utilisée afin de comparer le mot de passe saisi avec le hash stocké en base de données. Cette approche garantit qu'aucun mot de passe lisible ne puisse être exposé, même en cas d'accès non autorisé à la base de données.

```
if (!password_verify($password, $u['password_hash'])) return false;
```

Vérification sécurisée du mot de passe à l'aide de `password_verify`.

L'application repose également sur une gestion des rôles afin de contrôler l'accès aux fonctionnalités. Deux rôles sont définis : *CLIENT* et *ADMIN*. Les droits d'accès sont vérifiés côté serveur avant l'affichage ou l'exécution de toute action sensible. Les pages réservées à l'administration sont ainsi inaccessibles aux utilisateurs non autorisés, même en cas de tentative d'accès direct par l'URL.

```
function require_role(string $role): void {  
    require_login();  
    $u = current_user();  
    if (($u['role'] ?? null) !== $role) {  
        flash_set('error', 'Accès refusé.');
```

Contrôle d'accès basé sur le rôle de l'utilisateur (CLIENT / ADMIN)

Les formulaires de l'application sont protégés contre les attaques de type CSRF (Cross-Site Request Forgery). Chaque formulaire intègre un jeton de sécurité généré côté serveur et stocké en session. Ce jeton est vérifié lors de la soumission du formulaire afin de s'assurer que la requête provient bien de l'utilisateur authentifié et non d'une source externe malveillante.

```
function csrf_token(): string {
    if (empty($_SESSION['csrf'])) {
        $_SESSION['csrf'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf'];
}

function csrf_verify(?string $token): bool {
    return isset($_SESSION['csrf']) && is_string($token) && hash_equals($_SESSION['csrf'], $token);
}
```

Génération et vérification d'un jeton CSRF pour sécuriser les formulaires

L'ensemble des accès à la base de données est sécurisé à l'aide de requêtes préparées via PDO. Cette technique empêche l'injection de code SQL en séparant strictement les requêtes des données fournies par l'utilisateur. Les paramètres sont liés dynamiquement, ce qui garantit que les entrées utilisateur ne peuvent pas être interprétées comme du code SQL.

```
$stmt = $pdo->prepare("
    UPDATE appointment
    SET start_at = :start_at,
        end_at = :end_at,
        id_artist = :id_artist,
        id_service = :id_service
    WHERE id_appointment = :id
");
$stmt->execute([
    ':start_at' => $start->format('Y-m-d H:i:s'),
    ':end_at' => $end->format('Y-m-d H:i:s'),
    ':id_artist' => $id_artist_new,
    ':id_service' => $id_service_new,
    ':id' => $id_appointment,
]);
```

Utilisation de requêtes préparées afin de prévenir les injections SQL.

Enfin, des contrôles d'accès serveur sont appliqués afin de sécuriser la navigation dans l'application. Les pages sensibles sont protégées par des vérifications systématiques de l'état de connexion et du rôle de l'utilisateur. Toute tentative d'accès non autorisée entraîne une redirection vers une page appropriée, renforçant ainsi la sécurité globale de l'application.

L'ensemble de ces mécanismes permet de garantir un niveau de sécurité cohérent et adapté à une application de gestion de rendez-vous, en appliquant les bonnes pratiques du développement web côté serveur

11. Tests et validation

Afin de valider le bon fonctionnement de l'application et de garantir la cohérence des règles métier, une phase de tests a été réalisée tout au long du développement. Ces tests ont permis de vérifier à la fois les parcours utilisateurs (client et administrateur), la logique métier de calcul des disponibilités ainsi que les mécanismes de sécurité mis en place côté serveur.

Dans un premier temps, l'authentification a été testée afin de s'assurer que l'accès à l'application est correctement sécurisé. Les tests ont porté sur la connexion avec des identifiants valides et invalides, ainsi que sur le comportement de l'application en cas de session non active. Le contrôle d'accès par rôle a également été vérifié, notamment en s'assurant qu'un utilisateur de type CLIENT ne puisse pas accéder aux pages réservées à l'ADMIN, même en tentant un accès direct par l'URL.

Ensuite, la fonctionnalité de réservation a été testée du point de vue client. La sélection d'un artiste, d'une prestation et d'une date déclenche l'affichage des créneaux disponibles, ce qui a permis de vérifier que la durée des prestations est correctement prise en compte lors de la génération des créneaux. La création d'un rendez-vous a été testée afin de confirmer que celui-ci est bien enregistré avec le statut *PENDING* et qu'il apparaît dans la liste des rendez-vous à venir du client.

Une attention particulière a été portée à la logique métier de prévention des conflits. Des cas de test ont été mis en place pour simuler des rendez-vous existants et des périodes d'indisponibilité. L'objectif était de vérifier que les créneaux en conflit ne sont jamais proposés à l'utilisateur et que l'application refuse l'enregistrement d'une réservation lorsque le créneau vient d'être occupé entre-temps. Ce test permet de garantir la cohérence du planning même en cas de concurrence entre utilisateurs.

La partie administration a également fait l'objet de tests fonctionnels. Les actions de gestion (confirmer, refuser, annuler) ont été vérifiées afin de garantir que les statuts évoluent correctement et que l'interface reflète immédiatement les mises à jour. La fonctionnalité de modification d'un rendez-vous (changement de date, artiste ou prestation) a été testée afin de valider le recalcul automatique de l'heure de fin, ainsi que la présence des contrôles empêchant une modification entraînant un chevauchement ou une réservation sur une période d'indisponibilité.

Enfin, des tests de sécurité ont été réalisés pour vérifier la robustesse des traitements côté serveur. La protection CSRF a été validée en s'assurant qu'une requête POST sans jeton correct est rejetée. Les requêtes préparées ont été utilisées sur l'ensemble des actions critiques afin de prévenir les injections SQL. Ces validations garantissent que l'application reste sécurisée face aux principales attaques web courantes.

L'ensemble de ces tests a permis de confirmer la stabilité des fonctionnalités principales et la conformité de l'application aux exigences métier définies en amont.

12. Conclusion et évolutions possibles

La réalisation de ce projet m'a permis de mobiliser et de consolider l'ensemble des compétences attendues dans le cadre du Titre Professionnel Développeur Web et Web Mobile, aussi bien sur la partie front-end que sur la partie back-end. Il m'a offert l'opportunité de concevoir une application complète, depuis l'analyse des besoins jusqu'à la mise en œuvre technique, en passant par la modélisation des données et la définition des règles métier.

Sur le plan technique, ce projet m'a permis de mettre en pratique la conception d'interfaces utilisateur claires et fonctionnelles, ainsi que le développement de traitements dynamiques côté serveur. J'ai pu implémenter une base de données relationnelle cohérente, sécuriser l'accès aux données et développer une logique métier robuste, notamment pour le calcul des disponibilités et la gestion des rendez-vous. L'application repose sur des choix techniques adaptés, privilégiant la sécurité, la maintenabilité et la cohérence des données.

Ce projet m'a également permis de mieux appréhender les contraintes d'un contexte métier réel. La gestion des durées de prestations, des disponibilités des artistes, des statuts de rendez-vous et des validations administratives illustre une problématique concrète et réaliste, proche de situations rencontrées en entreprise. La centralisation de la logique métier côté serveur garantit la fiabilité du système et limite les risques d'incohérence.

Enfin, plusieurs évolutions pourraient être envisagées afin d'enrichir l'application. L'envoi de notifications par email permettrait d'informer automatiquement les clients lors de la confirmation ou de la modification d'un rendez-vous. L'intégration d'un calendrier graphique offrirait une meilleure visualisation du planning pour les utilisateurs et les administrateurs. Une gestion plus avancée des horaires, incluant par exemple des plages variables selon les artistes ou les jours, pourrait également être mise en place. Enfin, le développement d'une interface mobile dédiée constituerait une évolution naturelle afin d'améliorer l'accessibilité et l'expérience utilisateur.

Ce projet constitue ainsi une base solide et évolutive, démontrant ma capacité à concevoir, développer et sécuriser une application web répondant à des besoins métiers précis.