Spécification fonctionnelle — ePark (niveau Codes 5.0)

1. Objectif

L'application ePark permet la mise en relation entre particuliers pour la réservation de places de parc à l'heure. Elle gère sites, places, disponibilités horaires, réservations (créneaux d'1h), marges de tolérance (10/15/20 min), paiements, notifications et back-office.

2. Glossaire

- Site : lieu géographique regroupant des places.
- Place : emplacement unique au sein d'un site.
- Propriétaire : utilisateur proposant des places.
- Locataire : utilisateur réservant une place.
- Créneau horaire : intervalle d'au moins 60 minutes.
- Battement (marge) : 10/15/20 minutes ajoutées en fin de créneau.

3. Rôles & permissions

- Utilisateur (locataire) : recherche, réserve, paie, consulte historique.
- Propriétaire : crée site/place, définit disponibilité, valide réservations, reçoit paiements.
- Hybride : cumule les deux précédents.

```
# Spécification fonctionnelle — ePark (mise à jour 11-02-2026)

Cette spécification reprend le contenu fonctionnel principal et intègre les dernières déci

## Rappel des objectifs
ePark permet la mise en relation entre particuliers pour la réservation de places de parc

## Mises à jour récentes (11-02-2026)
- Interface : harmonisation palette et composants (boutons, navigation), contrastes renfor
- Logo : composant `application-logo` remplacé par l'image ePark.
- Place : ajout d'un delai d'annulation configurable (12h ou 24h) par le proprietaire.
- Place : suppression douce (soft delete) bloquee si reservations actives/a venir.
- Reservation : modification possible pour le locataire (en attente, non payee) + annulati
- Admin : page stats dediee (`/admin/stats`) avec top places et taux d'occupation.
- Helpers : `format_chf()` reste obligatoire pour l'affichage des montants.

## Points techniques importants
- Stocker les prix en centimes : `hourly_price_cents` (integer) pour éviter problèmes d'ar
- Notifications : exécution de la migration obligatoire sur l'environnement de production
- Helpers : s'assurer que `app/Support/helpers.php` est disponible au bootstrap pour évite
- Annulation : delai applique par place (`cancel_deadline_hours`).
- Soft delete : les places utilisees passent par `deleted_at`.

## Actions à exécuter après déploiement (checklist)
- Installer `rsync` sur l'environnement de déploiement (recommandé) ou lancer le script av
- Lancer les migrations : `php artisan migrate --force` (ou planifier via CI/CD).
- Nettoyer et reconstruire caches :
  - `php artisan view:clear`
  - `php artisan cache:clear`
  - `php artisan config:clear`
- Redémarrer les workers/queue/processus si présents (Supervisor/systemd).

## Récapitulatif modélisation & règles (extraits)
```

- Entités principales : `users`, `sites`, `places`, `reservations`, `payments`, `notificat
- Règle de conflit créneaux : existing.date_*debut* < *new.date*_fin AND existing.date_*fin* > *
- Battements : stockés dans `reservations.battement_minutes` (0 / 10 / 15 / 20).
- Annulation : autorisee avant `date_debut - cancel_deadline_hours`.
- Reservation modifiable si statut en_attente et paiement pending.

Déploiement & rollbacks (recommandation)
- Build des assets localement: `npm ci && npm run build` (Vite).
- Préparer dossier `_deploy` (scripts actuels le font), transférer via `rsync -az --delete
- Post-deploy : `php artisan migrate --force && php artisan cache:clear && php artisan vie
- Rollback : garder snapshot DB ou utiliser stratégie de migration réversible ; conserver

Tests & critères d'acceptation (prioritaires)
- Test unitaire: règles de conflit des réservations.
- Test d'intégration: webhook paiement + création Payment.
- E2E: parcours réservation complet (recherche → réservation → paiement → confirmation) su

Prochaines tâches recommandées
- Exécuter migrations en production (priorité haute).
- Installer rsync sur l'environnement de déploiement ou valider la procédure `--allow-scp`
- Automatiser post-deploy (migrations + cache clear) si CI/CD autorisé à exécuter ces comm

---

Mis à jour le : 2026-02-11