

Paris Tennis Dispo

Agrégation des créneaux libres et alertes d'annulation pour les courts parisiens

« Un court vient de se libérer. » Quand cette notification arrive, vous êtes probablement le seul à Paris à le savoir.

DOCUMENT

v1.0

PRODUIT

iOS 1.0.0

DATE

Août 2026

AUTRICE

Jiayue Liu

STATUT

Soumis à l'examen de l'App Store

Sommaire

01	Synthèse et positionnement	Ce qu'est ce produit, à qui il s'adresse, et pourquoi il mérite d'exister.
02	Le problème : combien d'essais pour une réservation ?	Trois irritants qui se cumulent. Aucun n'est réhibitoire seul ; ensemble, ils suffisent à faire renoncer.
03	Intuition centrale : deux batailles distinctes	Renoncer à la ruée n'est pas un repli. C'est ainsi qu'on a trouvé une bataille que personne ne livre.
04	Paysage concurrentiel	Nous ne concurrençons aucun acteur en place : la transaction se conclut toujours chez eux.
05	Utilisateurs et parcours	Pas « tous les joueurs de tennis », mais ceux dont l'emploi du temps est irrégulier et qui refusent d'y renoncer.
06	Vue d'ensemble de la solution	Trois onglets pour trois états : chercher activement, attendre passivement, régler ses préférences.
07	Points forts de conception • Interface	Chaque décision ici peut répondre à la question « pourquoi pas la solution évidente ? »
08	Points forts de conception • Interaction	Une bonne conception d'interaction se manifeste souvent par ce que l'utilisateur n'a pas remarqué.
09	Périmètre fonctionnel	P0 : « sans cela, le produit n'existe pas ». P1 : « cela l'améliore ». P2 : « plus tard ».
10	Décisions structurantes et arbitrages	La partie la plus utile d'un document produit n'est pas ce qui a été construit, mais pourquoi les alternatives ne l'ont pas été.
11	Modèle économique	La partie gratuite doit être réellement utile. La partie payante doit être réellement irremplaçable.
12	Flux de données et architecture	L'architecture sert deux contraintes : garder un coût faible, et ne pas peser sur les plateformes amont.
13	Indicateurs	S'il ne fallait suivre qu'un chiffre, ce serait « combien de réservations ont eu lieu grâce à nous ».
14	Feuille de route et risques	Les risques identifiés, et les parades déjà en place.

01

Synthèse et positionnement

Ce qu'est ce produit, à qui il s'adresse, et pourquoi il mérite d'exister.

Trouver rapidement à Paris un court de tennis libre correspondant à vos critères, et être prévenu en premier lorsqu'un créneau se libère — puis basculer en un geste vers la plateforme officielle pour réserver.

Le produit ne réserve pas à la place de l'utilisateur, ne stocke aucun identifiant tiers et ne prive personne d'une ressource publique. Il fait une seule chose — la découverte — et cherche à la faire mieux que tout ce qui existe.

En trois phrases

- 1. Le problème.** Le site municipal ne révèle la disponibilité qu'une fois le court, la date et le créneau déjà choisis. Trouver une séance jouable demande des dizaines d'allers-retours — et la plupart des joueurs doivent surveiller en parallèle une seconde plateforme, Anybuddy.
- 2. L'intuition.** La véritable opportunité n'est pas la ruée quotidienne à l'ouverture des créneaux, où des centaines de personnes actualisent en même temps. C'est le créneau qu'un joueur annule à une heure imprévisible : aléatoire, non surveillé, et pratiquement sans concurrence.
- 3. La solution.** Agréger les disponibilités publiques des deux plateformes en une liste filtrable unique, et les surveiller en continu afin qu'une annulation correspondant à vos critères déclenche une notification.

54

sites surveillés en continu (43 centres municipaux + $\approx$ 11 clubs Anybuddy)

202

courts municipaux individuels parmi eux

 $\approx$ 19 800

cellules court $\times$ date $\times$ créneau couvertes à chaque balayage

10 min

cycle de rafraîchissement, avec vérification en temps réel à la demande

Le positionnement en une ligne

Non pas un outil de réservation, mais un **outil d'asymétrie d'information**. La transaction reste sur la plateforme officielle. Ce que nous vendons, c'est de savoir trente secondes avant les autres.

Pourquoi ce positionnement est un choix délibéré

Le concept initial (v0.1) consistait à réserver automatiquement à la place de l'utilisateur. Après évaluation, j'ai moi-même abandonné cette direction :

Critère	v0.1 Réservation auto	v1.0 Agrégation + alertes
Stockage des mots de passe	Indispensable	Aucun
RGPD et risque lié aux données	Élevé, expertise juridique requise	Léger : coordonnées et jeton de notification
CGU des plateformes amont	Violation des clauses anti-automatisation	Données publiques uniquement

Critère	v0.1 Réservation auto	v1.0 Agrégation + alertes
Publication sur l'App Store	Refus probable	Catégorie établie, viable
Éthique des biens publics	Prive d'autres joueurs de leur chance	Accélère seulement l'information
Coût serveur	Une collecte par utilisateur	Une collecte centralisée, partagée

Le virage qui a fait sortir le projet d'une zone grise pour en faire un produit publiable.

Ce choix revient à renoncer à la ruée quotidienne. La section 03 explique pourquoi cette perte est bien plus faible qu'il n'y paraît.

02

Le problème : combien d'essais pour une réservation ?

Trois irritants qui se cumulent. Aucun n'est rédhibitoire seul ; ensemble, ils suffisent à faire renoncer.

Un • La disponibilité est cachée derrière le choix

Le site municipal fonctionne selon une logique « choisir puis révéler » : il faut d'abord désigner un court, une date et un créneau, valider, et seulement alors apprendre s'il reste quelque chose. **L'information de disponibilité se situe derrière l'acte de choisir, et non devant.**

Chercher une séance devient donc cette boucle :

1. Choisir un court
2. Choisir une date
3. Choisir un créneau
4. Valider et attendre le chargement
5. Lire « aucun court disponible »
6. Revenir en arrière et essayer un autre créneau

🔄 **Retour à l'étape 3 — chaque tentative est un aller-retour complet, et l'on ignore combien il en reste.**

Le coût de cette boucle se mesure. Dans le périmètre surveillé par ce produit :

43

centres de tennis municipaux

×

jours de fenêtre de réservation

301

allers-retours pour parcourir une seule fois une semaine de centres municipaux

≈ 19 800

202 courts × 7 jours × 14 créneaux : les cellules réellement en jeu

Le problème n'est pas la lenteur, c'est l'impossibilité d'être exhaustif

La question que se pose réellement l'utilisateur est : « **dans les sept prochains jours, quand y a-t-il un court en terre battue couvert et libre en soirée à moins de 5 km ?** » L'interface actuelle ne sait pas y répondre. Elle ne sait répondre qu'à « cette case est-elle libre ? ». L'utilisateur ne peut approcher la vraie réponse qu'à force de dizaines d'essais, sans jamais savoir s'il est passé à côté de mieux.

Deux • Deux plateformes, un va-et-vient permanent

L'offre de courts à Paris est scindée en deux systèmes qui ne communiquent pas :

Courts municipaux (tennis.paris.fr)

Tarif bas (environ 12 à 20 € l'heure), mais des courts très disputés aux heures prisées et limités aux équipements publics.

Anybuddy

Couvre les clubs privés, davantage de disponibilités, à un tarif trois à quatre fois supérieur.

Concrètement, quiconque joue une à deux fois par semaine **doit surveiller les deux** : tenter d'abord sa chance sur le municipal, puis se rabattre sur Anybuddy. Ce qui implique :

- Deux logiques de filtrage et deux langages d'interface à maintenir mentalement en parallèle
- Deux boucles d'essais-erreurs indépendantes — le coût du premier irritant, doublé
- **Aucune comparaison possible** : impossible de voir dans une même vue « mercredi soir, 20 € en municipal et 48 € en club »
- Une perte de contexte à chaque bascule : les critères posés d'un côté doivent être ressaisis de l'autre

Trois · Le tennis est dilué dans une application multisport

Anybuddy se positionne comme une **plateforme multisport de réservation et de mise en relation** — tennis, badminton, squash, pickleball, football — avec la recherche de partenaires comme usage central.

Ce positionnement est pertinent pour Anybuddy. Pour qui ne joue qu'au tennis, il signifie :

- Chaque ouverture de l'application commence par un filtrage « je veux du tennis » : une friction pure en usage fréquent
- **Les critères propres au tennis sont relégués.** La surface (terre battue / dur / gazon synthétique) et la couverture sont déterminantes pour un joueur de tennis et sans objet pour un joueur de squash — ils ne peuvent donc pas occuper le premier niveau d'une hiérarchie partagée
- Les fonctions sociales occupent une place centrale dont un joueur ayant déjà son partenaire n'a pas besoin

Notre angle

Extraire une couche verticale d'une plateforme horizontale et la reconstruire. Au niveau de l'implémentation, c'est littéral : le collecteur lit la sous-page `/tennis` de chaque club, si bien que les stocks de pickleball, de badminton et de squash n'entrent jamais dans nos données. Nous ne prenons que les données tennis des deux sources et réorganisons la hiérarchie autour de ce qui compte vraiment pour un joueur : distance, horaire, surface, couverture, prix, source. Des filtres enfouis ailleurs au troisième niveau deviennent ici des interrupteurs du premier écran.

L'effet cumulatif

Pris isolément, chaque irritant reste supportable. Mais ils se multiplient au lieu de s'additionner : coût d'une boucle d'essais × deux plateformes × un filtrage sport à chaque fois. Résultat : **le coût attentionnel pour trouver un court à Paris est assez élevé pour que beaucoup renoncent au jeu spontané** et ne conservent qu'un créneau fixe hebdomadaire.

Le produit ne cherche pas à supprimer un clic. Il cherche à supprimer la pensée « laisse tomber, pas aujourd'hui ».

03

Intuition centrale : deux batailles distinctes

Renoncer à la ruée n'est pas un repli. C'est ainsi qu'on a trouvé une bataille que personne ne livre.

« Obtenir un court prisé » semble être un seul problème. En le décomposant, on découvre **deux batailles totalement différentes**, dont l'intensité concurrentielle diffère d'un ordre de grandeur.

Bataille A • L'ouverture des créneaux

Le système municipal libère de nouveaux créneaux à heure fixe chaque jour. Des centaines de personnes actualisent simultanément et les séances prisées partent en quelques secondes.

- Concurrents : plusieurs centaines, en simultané
- Facteur décisif : latence réseau et rapidité — ou automatisation
- Conclusion : **sans réservation automatisée, cette bataille est perdue d'avance. Abandon assumé.**

Bataille B • Les créneaux libérés par annulation

Lorsqu'un joueur annule, le créneau redevient disponible. Les caractéristiques sont radicalement différentes :

Caractéristique	A • Ouverture	B • Annulations
Moment	Heure fixe, connue de tous	Aléatoire, à toute heure
Concurrents simultanés	Des centaines	Proche de zéro
Facteur décisif	Vitesse et latence	Qui l'apprend en premier
Couvrable humainement ?	Oui — une alarme suffit	Non. Personne n'actualise deux cents fois par jour
Couvrable par un serveur ?	Oui	Oui, sans effort

C'est là que réside l'avantage décisif

Un humain ne peut pas actualiser deux cents fois par jour. Un serveur peut balayer toute la ville toutes les dix minutes. « **Demain 20 h, un court en terre battue couvert à Élisabeth vient de se libérer** » — **quand cette notification arrive, vous êtes probablement le seul à Paris à le savoir.** Cette valeur n'est en rien inférieure à celle de la réservation automatique, et elle est parfaitement légitime.

Comment cette intuition a redéfini le produit

Dès lors que la bataille B est identifiée comme l'opportunité réelle, le centre de gravité passe de la recherche à la surveillance :

Si la recherche est centrale

Le produit est un meilleur annuaire. L'utilisateur l'ouvre, interroge, repart. La valeur croît linéairement avec la fréquence d'usage — laquelle est structurellement faible.

Si la surveillance est centrale

Le produit travaille pendant que l'utilisateur ne le regarde pas. La valeur se matérialise au moment de la notification — un moment qu'il n'aurait pas pu produire seul.

Cela a déterminé trois choses : **les alertes sont un onglet principal et non une fonction annexe ; la recherche reste gratuite à vie, seules les alertes sont payantes** (section 11) ; et **la qualité des notifications devient le principal problème d'ingénierie** (section 08).

04

Paysage concurrentiel

Nous ne concurrençons aucun acteur en place : la transaction se conclut toujours chez eux.

	tennis.paris.fr	Anybuddy	Paris Tennis Dispo
Périmètre	Courts municipaux	Clubs partenaires	Les deux, agrégés
Sports	Tennis et autres sports municipaux	Multisport	Tennis uniquement
Vue d'ensemble	Case par case	Liste, correcte	Liste unifiée inter-plateformes
Filtre surface / couvert	Limité	Enfoui	Interrupteurs du 1er écran
Alertes d'annulation	Aucune	Aucune	Fonction centrale
Comparaison de prix	Impossible	Stock propre uniquement	Sur le même écran
Réservation	Native	Native	Renvoi vers la plateforme officielle

Complémentaire, non substitutif

Le produit ne prend aucune commission, ne gère aucune transaction et ne stocke aucun identifiant amont. La découverte se fait ici, la réservation là-bas. C'est à la fois une nécessité de conformité et une manière d'éviter l'affrontement direct avec les plateformes dont nous dépendons.

Pourquoi cette position est défendable

- **Aucun acteur en place n'est incité à le faire.** Le système municipal ne subit aucune pression commerciale pour améliorer l'expérience ; Anybuddy n'a aucun intérêt à orienter ses utilisateurs vers le stock d'un concurrent.
- **L'agrégation constitue elle-même la barrière à l'entrée.** La valeur naît du fait de surveiller les deux côtés à la fois ; un produit mono-source ne peut pas la reproduire.
- **Les alertes accumulent un actif.** L'historique des variations fonde la capacité de prédiction future (section 14).
- **Profondeur verticale.** Ne servir que le tennis autorise un niveau de détail sur la surface, la couverture et l'orientation qu'une plateforme horizontale ne peut structurellement pas se permettre.

05

Utilisateurs et parcours

Pas « tous les joueurs de tennis », mais ceux dont l'emploi du temps est irrégulier et qui refusent d'y renoncer.

Persona principal

Critère	Description
Qui	Joueur amateur vivant à Paris, avec un ou deux partenaires réguliers
Fréquence	Souhaite jouer une à deux fois par semaine
Contrainte	Horaires de travail irréguliers, impossible de bloquer un créneau longtemps à l'avance ; sensible au prix sans être prêt à tout
Comportement actuel	Essais répétés sur le site municipal, repli sur Anybuddy ; renonce souvent parce que « c'est trop compliqué »
Besoin non satisfait	« Dites-moi quand c'est libre, au lieu de me faire demander case par case. »

Parcours A • Chercher un court maintenant

- 1 Ouvrir l'application, arriver directement sur « Trouver » — sans inscription ni connexion
- 2 Renseigner une adresse de départ (une seule fois), utilisée pour le filtre distance
- 3 Régler les filtres : distance / date / horaires / surface / couverture / source
- 4 Voir les résultats municipaux et Anybuddy dans une même liste, triable par distance, heure ou prix
- 5 Ouvrir une séance : nom du court, adresse, temps à vélo, prix, surface
- 6 Toucher « Réserver sur le site officiel » — les informations sont copiées automatiquement

Parcours B • Alertes d'annulation (cœur du produit)

- 1 Créer une ou plusieurs surveillances dans l'onglet « Alertes »
- 2 Définir « mes créneaux de jeu » — par jour de la semaine, une fois pour toutes
- 3 Le serveur balaie en continu ; toute correspondance déclenche une notification
- 4 La notification arrive → il suffit de la toucher pour ouvrir le détail de la séance
- 5 Renvoi vers la plateforme officielle pour réserver

Articulation des deux parcours

Le parcours A relève de **l'acquisition et de la confiance** : gratuit à vie, il permet d'éprouver une valeur réelle avant tout paiement. Le parcours B relève de **la valeur et de la monétisation** : il accomplit ce que l'utilisateur ne peut pas faire seul. Le premier prouve que les données sont vraies ; seul le second justifie un abonnement.

06

Vue d'ensemble de la solution

Trois onglets pour trois états : chercher activement, attendre passivement, régler ses préférences.

Onglet	Problème traité	Interaction centrale
Trouver	Irritants 1 à 3 : ramener près de vingt mille cellules de disponibilité à une liste filtrable	Six dimensions de filtre, liste unifiée, trois tris
Alertes	Bataille B : continuer à travailler quand l'utilisateur ne regarde pas	Conditions de surveillance + créneaux de jeu + notifications
Réglages	Adresse, langue, notifications, abonnement	Peu fréquent, mais toujours accessible

Un arbitrage structurant en architecture de l'information

« Trouver » et « Alertes » partagent le même vocabulaire de filtres — distance, horaires, surface, couverture, source — mais **ils maintiennent deux états indépendants**.

Pourquoi ne pas partager un seul état

« Ce que je cherche maintenant » et « ce dont je veux être prévenu » sont deux intentions différentes. La première est ponctuelle, la seconde durable. Les mutualiser laisserait une recherche anodine corrompre silencieusement une règle d'alerte de long terme.

Coût et compensation

Le coût est de configurer deux fois. La compensation : « Alertes » est volontairement plus simple — les créneaux de jeu se règlent une fois pour chaque jour de la semaine et restent valables indéfiniment.

07

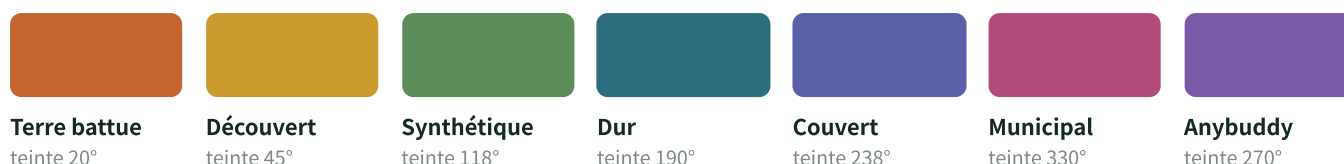
Points forts de conception • Interface

Chaque décision ici peut répondre à la question « pourquoi pas la solution évidente ? »

Un • Sept dimensions de filtre, sept couleurs, aucune collision

L'écran « Trouver » doit afficher simultanément six à sept groupes d'étiquettes : trois surfaces, couvert ou découvert, deux sources, plus les dates. Des couleurs choisies au hasard interféreraient et obligerait à lire chaque libellé.

La palette n'a donc pas été choisie étiquette par étiquette, mais **répartie sur la roue chromatique** :



- Les cinq premières sont à peu près équidistantes sur la roue : **deux étiquettes voisines diffèrent assez en teinte** pour être distinguées sans lecture ;
- Les deux couleurs de source ont été placées délibérément dans les intervalles restants — violet et rose — pour ne jamais entrer en conflit avec une couleur de surface ;
- **La sémantique se suffit à elle-même** : terre cuite pour la terre battue, vert gazon pour le synthétique, or solaire pour le découvert, indigo d'ombre pour le couvert. Ces couleurs n'ont pas à être mémorisées, elles s'expliquent d'elles-mêmes.

Les dates n'ont volontairement pas reçu de huitième teinte

La barre de dates utilise un vert sapin proche du texte courant plutôt qu'une teinte saturée supplémentaire — car **une date est un axe, pas une catégorie**. Elle n'est pas de même nature que les filtres situés en dessous. Lui attribuer une teinte laisserait croire qu'elle appartient à la même dimension que la surface ou la source. Exprimer une différence de niveau d'information par une différence de *nature* chromatique est plus efficace qu'ajouter un séparateur.

Deux • La couleur porte la reconnaissance, le texte ne fait que confirmer

Chaque carte de résultat porte une barre verticale colorée selon la surface. Lors d'un défilement rapide, **la vision périphérique suffit à écarter les surfaces indésirables** sans lire les libellés. L'étiquette textuelle subsiste, mais son rôle passe de la reconnaissance à la confirmation.

C'est un choix de hiérarchie assumé : avec vingt ou trente résultats à l'écran, faire passer le coût de reconnaissance de la *lecture* à la *perception* est la plus grande amélioration disponible pour une liste.

Trois • L'assombrissement du jaune tennis

La couleur de marque vient du jaune fluo de la balle. Sur le fond crème (#FAF8F3), ce fluo flotte, manque de contraste et devient illisible en petit corps. La valeur retenue est un #C9DA2E assombri d'un cran, avec un #8FA318 plus profond réservé aux contextes exigeant davantage de contraste.

Quand identité de marque et lisibilité s'opposent, la lisibilité l'emporte — mais la teinte étant conservée, rien n'est perdu de la marque.

Quatre · Le paywall montre le produit, pas des adjectifs

Le premier élément de l'écran d'abonnement n'est pas une liste de fonctionnalités, mais **une carte qui ressemble exactement à une vraie notification**. Avant de décider de payer, l'utilisateur voit précisément ce qu'il achète.

Parce qu'elle paraît vraie, la mention « Exemple » ne peut pas être omise

La carte porte une mention « Exemple » explicite. Sans elle, quelqu'un croirait qu'un court s'est réellement libéré et se déplacerait pour rien — **le réalisme achète la confiance, et cette confiance ne doit pas être monnayée par une tromperie**. La même mention, avec la même formulation, figure sur la carte d'exemple de l'onglet « Alertes ».

Cinq · Un détail : le curseur de l'interrupteur est une balle de tennis

Tout le monde a déjà vu l'interrupteur système par défaut. Remplacer son curseur par une balle de tennis est la seule modification purement décorative de toute l'application. Elle n'améliore aucun indicateur — mais elle fait que le produit ressemble à **quelque chose que quelqu'un a fait avec soin**, et non à un assemblage de composants.

Points forts de conception • Interaction

Une bonne conception d'interaction se manifeste souvent par ce que l'utilisateur n'a pas remarqué.

Un • Aucune inscription : le premier écran est le produit

L'application s'ouvre directement sur « Trouver » : on peut chercher immédiatement. Ni inscription, ni connexion, ni tutoriel. L'identifiant utilisateur est une chaîne aléatoire générée localement au premier lancement, sans aucune donnée personnelle.

Raison : le produit démontre sa valeur en trente secondes (« tiens, il y a vraiment un court libre à cette heure-là »). Tout ce qu'on place devant ces trente secondes impose un coût avant que l'utilisateur n'ait la moindre raison de nous faire confiance.

Deux • La recherche est gratuite à vie, seules les alertes sont payantes

Ce n'est pas qu'une politique tarifaire, c'est un choix de conception au service de la confiance : **l'utilisateur peut vérifier, sans rien payer, que nos données sont réelles, fraîches et exactes**. On ne parle d'argent qu'au moment où il veut que l'application surveille à sa place.

Le paywall l'affiche au lieu de le dissimuler : « La recherche reste gratuite. Les alertes sont sur abonnement. »

Trois • La fraîcheur des données est rendue explicite

Le principal risque de confiance pour un agrégateur est « ce que vous me montrez est-il périmé ? ». Le produit choisit de le dire ouvertement :

- Chaque jeu de résultats porte la mention « **mis à jour il y a X minutes** » ; la latence n'est pas masquée
- Une action « **Actualiser maintenant** » permet de forcer une vérification en temps réel au moment décisif
- Cette action possède une **temporisation**, pour que personne ne croie qu'appuyer plusieurs fois améliore la précision
- Le texte précise que les données proviennent des pages publiques officielles et que **la page officielle fait foi**

Reconnaître une limite construit la confiance

Inscrire « il peut y avoir quelques minutes de décalage » dans l'interface vaut mieux que prétendre à un temps réel parfait. Si l'utilisateur rencontre une donnée périmée sans avoir jamais été prévenu, la confiance s'effondre d'un coup. S'il a été prévenu, le même incident est perçu comme normal.

Quatre • Réversibilité : l'adresse peut être effacée

L'adresse de départ est la seule information personnelle saisie. Une version antérieure permettait de l'enregistrer mais pas de la supprimer : le bouton était désactivé sur un champ vide, si bien qu'une fois stockée, l'adresse ne pouvait plus disparaître sans réinstaller l'application.

Vider le champ transforme désormais le bouton en « Effacer ». **Tout ce qui peut être saisi doit pouvoir être révoqué** — a fortiori s'il s'agit d'une donnée personnelle et que l'utilisateur peut tendre son téléphone à quelqu'un.

Cinq · Les horaires d'alerte suivent « quand je peux réellement jouer »

Les plages horaires ne se configurent pas alerte par alerte. Elles sont liées à un réglage durable, « mes créneaux de jeu », défini par jour de la semaine et valable indéfiniment.

Raison : les moments où l'on peut jouer découlent du rythme professionnel et personnel, stable à l'échelle de plusieurs mois. Faire ressaisir un fait stable à chaque création d'alerte revient à reporter la paresse du système sur l'utilisateur.

Six · Les avertissements n'apparaissent que lorsqu'ils sont pertinents

Sur les 43 centres municipaux parisiens et leurs 202 courts, **deux exactement sont en terre battue**. Lorsqu'un utilisateur coche « terre battue », une explication s'impose, sans quoi il croira l'application défaillante.

Mais cet avertissement **n'apparaît que si la source municipale et la terre battue sont sélectionnées ensemble**. L'afficher à quelqu'un qui ne regarde même pas les courts municipaux n'est que du bruit.

Un avertissement affiché dans les bonnes conditions est une aide ; le même dans les mauvaises est une gêne. Plus la condition est écrite finement, plus l'avertissement a de valeur.

Sept · La carte d'exemple disparaît une fois sa mission accomplie

L'onglet « Alertes » s'ouvre sur une carte « voici à quoi ressemble une notification », qui explique ce que la fonction va réellement apporter. Dès que l'utilisateur **a reçu une vraie alerte d'annulation**, cette carte disparaît définitivement — et cet état ne revient jamais en arrière.

Raison : un contenu pédagogique s'adresse à qui ne sait pas encore. Celui qui a déjà reçu la vraie notification n'a pas besoin qu'elle occupe un écran. Les notifications de test ne comptent pas : elles ne prouvent rien du fonctionnement réel de la chaîne.

Huit · Défaillance permissive : mieux vaut laisser passer que verrouiller un abonné

Le statut d'abonnement est interrogé localement auprès de StoreKit. En cas d'échec — coupure réseau, anomalie système — le produit **accorde l'accès** plutôt que de le bloquer.

Si l'échec bloque

Une simple coupure réseau exclut un client payant de ce qu'il a acheté, et il en conclut que son abonnement est défaillant. Ce mode de défaillance est inacceptable.

Si l'échec autorise

Dans de rares cas, un non-abonné aperçoit brièvement du contenu payant. Le coût est bien moindre, et en version de production la probabilité d'échec est très faible.

Neuf · Les informations sont copiées au moment du renvoi

Toucher « Réserver sur le site officiel » copie le nom du court et le créneau dans le presse-papiers, et affiche un guide visuel des étapes suivantes sur l'autre plateforme. C'est **une conception qui assume sa frontière** : nous ne maîtrisons pas leur interface, mais nous pouvons réduire au minimum le coût du franchissement.

09

Périmètre fonctionnel

P0 : « sans cela, le produit n'existe pas ». P1 : « cela l'améliore ». P2 : « plus tard ».

P0 • Indispensable au lancement

#	Fonction	Précisions
F1	Adresse de départ	Géocodée pour la distance ; modifiable et effaçable
F2	Filtre distance	1 / 3 / 5 / 10 / 20 / 40 km
F3	Filtre date et horaires	7 jours à venir ; plage horaire ajustable
F4	Filtre couvert / découvert	Deux interrupteurs indépendants
F5	Filtre surface	Terre battue / dur / gazon synthétique
F6	Filtre source	Municipal / Anybuddy / les deux
F7	Liste unifiée	Tri par distance, heure ou prix ; groupée par source
F8	Détail et renvoi	Détail + lien officiel + copie auto + guide
F9	Alertes d'annulation	Notification dès qu'une correspondance apparaît ; surveillances multiples
F10	Courts favoris	Suivre les courts où l'on joue le plus souvent
F11	Fraîcheur des données	« Mis à jour il y a X min » + actualisation + temporisation
F12	Interface trilingue	Français / anglais / chinois simplifié ; suit le système, réglable manuellement
F13	Abonnement et restauration	Mensuel / annuel + essai gratuit + restauration des achats

P1 • Deuxième version

- Vue carte avec points et cercles de distance
- Enregistrement de combinaisons de filtres en préréglages
- Météo : probabilité de pluie signalée sur les créneaux des courts découverts
- Comparaison de prix entre plateformes pour un même horaire
- Historique : « ce court était libre trois des quatre derniers mercredis à 20 h »

P2 • Plus tard

- Version Android
- Notifications Apple Watch et widgets
- Autres villes, ou autres sports de raquette (padel / squash)
- Prédiction de disponibilité à partir de l'historique

Volontairement exclu

La réservation automatique (conformité et éthique), **la mise en relation de partenaires** (hors du cœur de valeur, et terrain d'Anybuddy), **un système de comptes** (le produit fonctionne sans ; avec, il faudrait assumer la responsabilité des données). Chaque exclusion est une décision, non un oubli.

10

Décisions structurantes et arbitrages

La partie la plus utile d'un document produit n'est pas ce qui a été construit, mais pourquoi les alternatives ne l'ont pas été.

Un • Filtrer à vol d'oiseau, afficher le temps à vélo

Décision

Le filtrage utilise la distance à vol d'oiseau (gratuite, instantanée). L'API cartographique n'est appelée que pour les quelques séances que l'utilisateur ouvre réellement.

Raison

La Seine et le réseau ferré rendent la distance à vol d'oiseau imprécise à Paris. Mais « 1/3/5/10 km » reste un palier grossier et l'écart y demeure contenu, tandis que le temps à vélo précis ne compte que pour les deux ou trois derniers candidats. Le calcul à la demande divise l'usage de l'API par cent.

Deux • Collecte centralisée plutôt que par utilisateur

Décision

Le serveur collecte une fois ; tous les utilisateurs partagent le même instantané.

Raison

Coût et courtoisie. Une collecte par utilisateur ferait croître le volume de requêtes proportionnellement au nombre d'utilisateurs, augmentant le coût et pesant inutilement sur les sites amont. La centralisation découple le volume de requêtes de la croissance.

Trois • Le droit d'accès est vérifié sur l'appareil ; le serveur ne voit aucun reçu

Le statut d'abonnement est interrogé directement auprès de StoreKit — la réponse est produite localement et signée par Apple. En l'absence d'abonnement, le produit **révoque côté serveur le jeton de notification** : le serveur ne peut alors plus rien envoyer à cet appareil.

Cette conception supprime toute une catégorie de travail

Le serveur n'a jamais besoin de savoir qui a payé. Donc **aucune validation de reçu, aucun service tiers de gestion d'abonnements, aucune donnée de paiement stockée**. La barrière d'accès n'est pas une condition dans le code : c'est le fait physique qu'un jeton existe ou non. Le jeton révoqué, les notifications cessent par construction.

Quatre • Prix et durée d'essai lus depuis StoreKit à l'exécution

Rien sur le paywall — prix, périodicité, durée d'essai — n'est écrit en dur. Chaque valeur est interrogée auprès de StoreKit au chargement de l'écran.

Raison : la tarification d'Apple, les taux de change et la fiscalité diffèrent selon les pays ; toute valeur écrite en dur finit par devenir fausse quelque part — et « le prix affiché ne correspond pas au montant débité » est un motif de

refus explicite. De même, si aucun essai gratuit n'est configuré, l'interface ne mentionne aucune gratuité : le bouton devient « S'abonner » et la mention de reconduction bascule sur sa variante sans essai.

Cinq · Aucun système de comptes

L'identifiant utilisateur est une chaîne aléatoire générée au premier lancement. Changer de téléphone fait donc perdre à l'utilisateur sa configuration — un coût réel.

Il est accepté parce qu'un système de comptes apporte tout un cortège de charges — friction à l'inscription, récupération de mot de passe, demandes de suppression, obligations RGPD, alors que **le cœur de valeur — recevoir une notification au bon moment — n'en a aucun besoin**. Lorsque le coût peut être énoncé clairement et que le bénéfice est manifestement inférieur à la charge, la bonne réponse est de ne pas le construire.

11

Modèle économique

La partie gratuite doit être réellement utile. La partie payante doit être réellement irremplaçable.

	Gratuit	Abonnement
Recherche et filtres inter-plateformes	Gratuit à vie	Inclus
Détail et renvoi vers la réservation	Gratuit à vie	Inclus
Courts favoris	Gratuit à vie	Inclus
Alertes d'annulation	—	Avantage central
Surveillances multiples	—	Inclus

Tarification

1,99 €

par mois

12,99 €

par an — soit 45 % de moins que le mensuel

7 jours

d'essai gratuit

La référence est une réservation municipale (environ 12 à 20 €). **Le tarif annuel représente moins d'une séance et demie**, un ancrage qui dispense presque de calculer : une réservation supplémentaire dans l'année, obtenue grâce à une alerte, rentabilise l'abonnement.

Pourquoi la recherche reste gratuite à vie

- **La vérification passe avant.** L'utilisateur confirme que les données sont réelles avant de payer — le principal obstacle de confiance pour tout agrégateur ;
- **Acquisition.** Une recherche inter-plateformes gratuite mérite en soi d'être partagée et recommandée ;
- **Séparation nette de la valeur.** Le gratuit, c'est « je pourrais le faire moi-même, en plus lent ». Le payant, c'est « je ne pourrais pas le faire du tout ». Seul le second soutient un abonnement.

12

Flux de données et architecture

L'architecture sert deux contraintes : garder un coût faible, et ne pas peser sur les plateformes amont.

Chaîne de traitement

- 1 **Collecte** — le serveur lit les pages publiques de disponibilité des deux sources, environ toutes les 10 minutes
- 2 **Normalisation** — deux schémas différents deviennent un modèle unique : court, coordonnées, horaire, prix, surface, couverture, source
- 3 **Stockage** — instantané courant et historique des variations ; c'est l'historique qui rend la détection d'annulation possible
- 4 **Appariement** — les créneaux nouvellement libres sont croisés avec les conditions de tous les utilisateurs
- 5 **Notification** — envoi via le jeton, avec un plafond quotidien par utilisateur

Techniquement, une annulation est une différence, pas une requête

La fonction centrale n'est pas « interroger si un court est libre », mais « **détecter un créneau qui vient de passer d'indisponible à disponible** ». C'est une différence entre deux instantanés, pas une requête sur un seul. Cette définition impose de conserver l'historique, et fait de la fréquence de collecte un paramètre produit et non un détail technique.

Client

Couche	Choix
Cadre technique	React Native (Expo) ; compilation et soumission via EAS
État et persistance	Stockage local, aucun compte distant
Abonnements	StoreKit 2 via expo-iap ; droit d'accès décidé entièrement sur l'appareil
Notifications	Jeton enregistré côté serveur, révoqué à l'expiration du droit
Internationalisation	Trois dictionnaires ; suit la langue du système, modifiable manuellement

Minimisation des données

- **Non collecté** : nom, adresse e-mail, téléphone, identifiants tiers
- **Transmis** : coordonnées du point de départ (appariement par distance), jeton de notification, identifiant aléatoire, préférences de filtre
- **Non conservé** : l'adresse textuelle ne sert qu'au géocodage en temps réel et n'est jamais stockée
- **Aucun pistage** : pas de SDK publicitaire, aucun outil de mesure d'audience, aucun suivi inter-applications ou inter-sites

13

Indicateurs

S'il ne fallait suivre qu'un chiffre, ce serait « combien de réservations ont eu lieu grâce à nous ».

Indicateur phare

Réservations engagées depuis une notification

Il est retenu parce qu'il valide simultanément quatre choses : la donnée était juste, l'appariement était pertinent, le moment était utile, et l'utilisateur était prêt à agir. Si un maillon cède, le chiffre chute. Téléchargements, rétention et nombre d'abonnés peuvent tous paraître sains alors que le produit ne crée aucune valeur.

Indicateurs de soutien

Étape	Indicateur	Ce qu'il surveille
Découverte	Taux de clic recherche → détail	Pertinence des résultats
Surveillance	% d'utilisateurs avec une surveillance active	Compréhension et adoption de la fonction centrale
Notification	Taux d'ouverture des notifications	Densité d'information du message
Qualité	Conversion notification → renvoi	Envoie-t-on des créneaux non désirés ?
Monétisation	Conversion essai → payant	La valeur se matérialise-t-elle en 7 jours ?
Rétention	Taux de reconduction	Valeur durable ou ponctuelle ?

Le chiffre à protéger est la conversion notification → renvoi

Le mode de défaillance dominant n'est pas « trop peu de notifications », c'est « **beaucoup de notifications, aucune pertinente** ». Face à une application qui interrompt sans cesse et aide rarement, l'utilisateur fait exactement une chose : révoquer l'autorisation de notification. Une fois l'autorisation révoquée, le cœur de valeur tombe à zéro et la reconquête est très improbable. D'où un plafond quotidien strict : mieux vaut en manquer une que d'en envoyer trop.

14

Feuille de route et risques

Les risques identifiés, et les parades déjà en place.

Feuille de route

Phase	Priorité	Critère de réussite
Maintenant • v1.0	Publier ; valider que les alertes créent une valeur réelle	Un utilisateur réserve grâce à une notification
Ensuite • v1.1	Vue carte, préreglages de filtres, météo	Le taux de clic recherche → détail progresse
Puis • v1.2	Prédiction de disponibilité à partir de l'historique	La conversion notification → renvoi progresse
Plus tard	Autres villes / sports de raquette / Android	Le modèle se reproduit dans une deuxième ville

Risques et parades

Risque	Impact	Parade
Modification des pages amont, collecte mise en échec	Élevé — flux interrompu	Normalisation découplée de la collecte ; la fraîcheur étant visible, une panne se manifeste au lieu de rester silencieuse
Blocage de la collecte par l'amont	Élevé	La centralisation minimise déjà le volume ; données publiques uniquement, sans connexion ni réservation, ce qui laisse place au dialogue
Révocation de l'autorisation de notification	Élevé — valeur nulle	Plafond quotidien strict ; conversion notification → renvoi suivie comme indicateur avancé
Annulations rares en pratique	Moyen — cœur de valeur mince	La surveillance de 43 centres municipaux, soit 202 courts, élargit l'échantillon ; le stock Anybuddy complète l'offre
Plafond d'une ville unique	Moyen	La ville est un paramètre de configuration, non une hypothèse figée dans le code

Pour conclure

*Ce que ce produit démontre en définitive, c'est une chose modeste : dans un domaine où l'information était déjà publique, **la réorganiser suffit à créer une valeur réelle**. L'utilisateur paie toujours la plateforme officielle ; les courts appartiennent toujours à la ville. La seule chose qui change, c'est qu'il sait enfin quand l'un d'eux est libre.*

Paris Tennis Dispo

Cross-platform court availability and cancellation alerts for Paris

"A court just opened up." When that notification fires, you are probably the only person in Paris who knows.

DOCUMENT

v1.0

PRODUCT

iOS 1.0.0

DATE

August 2026

AUTHOR

Jiayue Liu

STATUS

Submitted for App Store review

Contents

01	Summary and Positioning	What this product is, who it serves, and why it deserves to exist.
02	The Problem: How Many Times Must You Guess?	Three pain points that compound. None is fatal alone; together they are enough to make people stop playing.
03	Core Insight: Two Different Battles	Giving up the rush is not a retreat. It is how we found a battle nobody is fighting.
04	Competitive Landscape	We compete with neither incumbent — the transaction still closes on their platform.
05	Users and Journeys	Not everyone who plays tennis — people whose schedule is irregular but who refuse to give it up.
06	Solution Overview	Three tabs for three states: actively looking, passively waiting, managing yourself.
07	Design Highlights • Interface	Every decision here can answer the question "why not the obvious alternative?"
08	Design Highlights • Interaction	Good interaction design often shows up as something the user never noticed being handled.
09	Scope	P0 is "without this it does not exist". P1 is "this makes it better". P2 is "later".
10	Key Decisions and Trade-offs	The most useful part of a product document is usually not what was built, but why the alternatives were not.
11	Business Model	The free tier has to be genuinely useful. The paid tier has to be genuinely irreplaceable.
12	Data Flow and Architecture	The architecture serves two constraints: keep cost low, and do not burden upstream.
13	Metrics	If only one number could be watched, it should be "how many bookings happened because of us".
14	Roadmap and Risks	Known risks, and the mitigations already built in.

01

Summary and Positioning

What this product is, who it serves, and why it deserves to exist.

Find an available tennis court in Paris that matches your constraints — and be the first to know when someone cancels. Then tap through to the official platform to book it.

The product does not book on the user's behalf, never stores third-party credentials, and does not compete for public resources. It does one thing — discovery — and tries to do it better than anything else available.

Three sentences

1. **The problem.** Paris's municipal booking site only reveals availability after you have already chosen a court, a date and a time slot. Finding one workable session takes dozens of trial-and-error round trips — and most players have to watch a second platform, Anybuddy, at the same time.
2. **The insight.** The real opportunity is not the daily release rush, where hundreds of people refresh simultaneously. It is the slot someone cancels at a random hour — unpredictable, unwatched, and essentially uncontested.
3. **The solution.** Aggregate publicly visible availability from both platforms into one filterable list, and monitor it continuously so a matching cancellation reaches the user as a push notification.

54

venues monitored continuously (43 municipal centres + ~11 Anybuddy clubs)

202

individual municipal courts among them

≈ 19,800

court × date × slot
availability cells covered by every scan

10 min

refresh cycle, with a user-triggered real-time re-check

Positioning in one line

Not a booking tool — an **information-asymmetry tool**. The transaction still happens on the official platform. What we sell is knowing thirty seconds before everyone else.

Why this positioning was a deliberate choice

The original concept (v0.1) was automated booking on the user's behalf. After evaluating it, I reversed course myself:

Dimension	v0.1 Auto-booking	v1.0 Aggregate + Alerts
Stores user passwords	Required	Never
GDPR and data risk	High — needs legal review	Light: coordinates and push token only
Upstream terms of service	Clearly violates anti-automation clauses	Publicly visible data only
App Store viability	Likely rejected	Established category, viable
Public-resource ethics	Crowds out other players	Only moves information faster

Dimension	v0.1 Auto-booking	v1.0 Aggregate + Alerts
Server cost	One crawl per user	One central crawl, shared by all

The pivot that moved the project out of a grey area and into something shippable.

The trade-off gives up the ability to win the daily release rush. Section 03 explains why that loss is much smaller than it appears.

02

The Problem: How Many Times Must You Guess?

Three pain points that compound. None is fatal alone; together they are enough to make people stop playing.

One • Availability is hidden behind the choice

The municipal booking site uses a choose-then-reveal structure: you must specify a court, a date and a slot, submit, and only then learn whether anything is free. **Availability sits behind the act of choosing rather than in front of it.**

Finding a session therefore becomes this loop:

1. Pick a court
2. Pick a date
3. Pick a time slot
4. Submit and wait for the page
5. Read "no courts available"
6. Go back and try a different slot

↻ Return to step 3 — every attempt is a full page round trip, and you never know how many more are left.

The cost of that loop is measurable. Within the scope this product monitors:

43

municipal tennis centres

×

days of booking window

301

page round trips to see one week of municipal courts — once

≈ 19,800

202 courts × 7 days × 14 slots: the availability cells actually at stake

The problem is not that it is slow — it is that it cannot be exhausted

The question a user actually has is: "**over the next seven days, when is there a covered clay court free in the evening within 5 km of me?**" The existing interface cannot answer that question. It can only answer "is this one cell free?" Users are forced to approximate the real answer through dozens of guesses, never sure whether they missed something better.

Two • Two platforms, constant switching

Court supply in Paris is split across two systems that do not talk to each other:

Municipal courts (tennis.paris.fr)

Cheap (roughly €12–20/hour) but heavily contested at popular hours, and limited to public facilities.

Anybuddy

Covers private clubs with more availability, at three to four times the price.

In practice, anyone playing once or twice a week **has to watch both** — try the municipal site first, fall back to Anybuddy. That means:

- Two different filtering models and interface languages to hold in your head at once
- Two independent trial-and-error loops — the cost of pain point one, doubled
- **No side-by-side comparison.** You cannot see "for Wednesday evening, here is €20 municipal and here is €48 private" in a single view
- Context is lost in the switch: constraints set on one platform must be re-entered on the other

Three · Tennis is diluted inside a multi-sport app

Anybuddy is positioned as a **multi-sport booking and social platform** — tennis, badminton, squash, pickleball, football — with partner-matching as a core use case.

That is the right positioning for Anybuddy. For someone who only plays tennis, it means:

- Every session begins with an "I want tennis" filtering action — pure friction at high frequency
- **Tennis-specific dimensions are de-emphasised.** Surface (clay / hard / synthetic grass) and covered-versus-open are decisive for a tennis player and meaningless to a squash player, so they cannot sit at the top of a shared information hierarchy
- Social features occupy prime interface real estate that a player with a regular partner does not need

Our angle

Take one vertical layer out of a horizontal platform and rebuild it. This is literal at the implementation level — the crawler reads each club's `/tennis` sub-page, so pickleball, badminton and squash inventory never enters our data at all. We pull only tennis data from both sources and reorganise the hierarchy around what tennis players actually weigh — distance, time, surface, cover, price, source. Filters buried three levels deep in a general platform become switches on the first screen.

The compounding effect

Each pain point is individually tolerable. They multiply rather than add: the cost of one trial-and-error loop × two platforms × a sport filter every time. The result is that **the attention cost of finding a court in Paris is high enough that many people abandon spontaneous play entirely** and keep only a fixed weekly slot.

The product is not trying to eliminate a click. It is trying to eliminate the thought "never mind, not today."

03

Core Insight: Two Different Battles

Giving up the rush is not a retreat. It is how we found a battle nobody is fighting.

"Getting a popular court" looks like one problem. Unpacked, it is **two entirely different battles**, an order of magnitude apart in competitive intensity.

Battle A • The release rush

The municipal system releases new bookable slots at a fixed time each day. Hundreds of people refresh at once and popular sessions are gone within seconds.

- Competitors: hundreds, simultaneously
- Decided by: network latency and reaction speed — or automation
- Conclusion: **without automated booking, this battle cannot be won. Deliberately abandoned.**

Battle B • Slots freed by cancellations

When someone cancels, that slot returns to the system as available. Its characteristics are completely different:

Characteristic	A • Release rush	B • Cancellations
When it happens	Fixed daily time, publicly known	Random, any hour, unpredictable
Simultaneous competitors	Hundreds	Close to zero
Decided by	Speed and latency	Who finds out first
Can a human cover it?	Yes — set an alarm	No. Nobody refreshes a page 200 times a day
Can a server cover it?	Yes	Yes, easily

This is the product's decisive advantage

A person cannot refresh two hundred times a day. A server can sweep the entire city every ten minutes.

"Tomorrow, 8pm, covered clay at Élisabeth just opened up" — when that notification arrives, you are probably the only person in Paris who knows. That is worth no less than automated booking, and it is entirely legitimate.

How the insight reshaped the product

Once Battle B was identified as the real opportunity, the centre of gravity moved from search to monitoring:

If search is the centre

The product is a better directory. Users open it, query, and leave. Value scales linearly with usage frequency — and that frequency is inherently low.

If monitoring is the centre

The product works while the user is not looking at it. Value is delivered at the moment the notification fires — a moment the user could not have produced alone.

That decided three things: **alerts are a primary tab, not a secondary feature; search stays free forever and only alerts are paid** (Section 11); and **notification quality becomes the single most important engineering problem** (Section 08).

04

Competitive Landscape

We compete with neither incumbent — the transaction still closes on their platform.

	tennis.paris.fr	Anybuddy	Paris Tennis Dispo
Coverage	Municipal courts only	Partner clubs only	Both, aggregated
Sports	Municipal facilities	Multi-sport	Tennis only
Availability at a glance	Cell-by-cell	List view, decent	Unified cross-platform list
Filter by surface / cover	Limited	Buried	First-screen switches
Cancellation alerts	None	None	Core feature
Price comparison	Impossible	Own inventory only	Same screen
Booking	Native	Native	Hand-off to the official site

Complementary, not a substitute

The product takes no commission, owns no transaction, and stores no upstream credentials. Discovery happens here; booking happens there. This is both a compliance necessity and a way to avoid a direct fight with the platforms we depend on.

Why this position is defensible

- **Neither incumbent is motivated to build it.** The municipal system faces no commercial pressure to improve UX; Anybuddy has no reason to route users towards a competitor's inventory.
- **Aggregation is itself the moat.** The value comes from watching both sides at once — a single-source product cannot reproduce it.
- **Alerts compound over time.** Accumulated change history becomes the basis for predicting availability (Section 14).
- **Vertical depth.** Serving only tennis allows a level of detail on surface, cover and orientation that a horizontal platform structurally cannot afford.

05

Users and Journeys

Not everyone who plays tennis — people whose schedule is irregular but who refuse to give it up.

Primary persona

Dimension	Description
Who	Amateur tennis player living in Paris, with one or two regular partners
Frequency	Wants to play once or twice a week
Constraint	Irregular work hours; cannot lock in slots far in advance. Price-sensitive but not at any cost
Today's behaviour	Repeated guessing on the municipal site, then Anybuddy as fallback; frequently gives up because "it is too much hassle"
Unmet need	"Tell me when something is free, instead of making me ask one cell at a time."

Journey A • Looking for a court now

- 1 Open the app, land directly on Find — no sign-up, no login
- 2 Set a home address once, used for distance filtering
- 3 Set filters: distance / date / hours / surface / cover / source
- 4 See municipal and Anybuddy results in one list, sortable by distance, time or price
- 5 Open a session for detail: court name, address, cycling time, price, surface
- 6 Tap "Book on the official site" — details are copied to the clipboard automatically

Journey B • Cancellation alerts (core)

- 1 Create one or more monitors on the Alerts tab
- 2 Define "when I can actually play" — set per weekday, once
- 3 The server keeps scanning; a match triggers a push immediately
- 4 Notification arrives → tap straight into the session detail
- 5 Hand off to the official platform to book

How the two journeys relate

Journey A is **acquisition and trust** — free forever, so users experience real value before paying. Journey B is **value delivery and monetisation** — it does something the user genuinely cannot do alone. The first proves the data is real; only the second is worth paying for.

06

Solution Overview

Three tabs for three states: actively looking, passively waiting, managing yourself.

Tab	Problem solved	Core interaction
Find	Pain points 1–3: compress nearly twenty thousand availability cells into one filterable list	Six filter dimensions, unified results, three sort orders
Alerts	Battle B: keep working while the user is not looking	Monitor conditions + playable hours + push
Settings	Address, language, notifications, subscription	Low frequency, but must stay reachable

One decisive information-architecture call

Find and Alerts share the same filter vocabulary — distance, hours, surface, cover, source — but **they hold two independent sets of state**.

Why not share one set

"What am I looking for right now" and "what do I want to be told about" are different intents. The first is momentary; the second is durable. Sharing state would let a casual search silently corrupt a long-running alert rule.

Cost and compensation

The cost is configuring twice. The compensation is that Alerts is deliberately simpler: playable hours are set once per weekday and stay valid indefinitely.

07

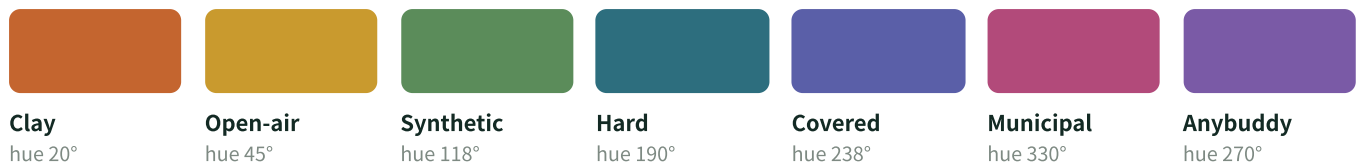
Design Highlights · Interface

Every decision here can answer the question "why not the obvious alternative?"

One · Seven filter dimensions, seven colours, no collisions

The Find screen must show six to seven groups of category chips at once: three surfaces, covered versus open, two sources, plus dates. Colours picked ad hoc would interfere on a shared screen and force users to read every label.

So the palette was not chosen chip by chip — it was **spaced around the hue wheel**:



- The first five are roughly equidistant on the wheel, so **any two adjacent chips differ enough in hue** to be told apart without reading;
- The two source colours were placed deliberately in the remaining gaps — violet and rose — so they never collide with a surface colour;
- **The semantics carry themselves**: terracotta for clay, grass green for synthetic, sunlit gold for open-air, shaded indigo for covered. The colours do not have to be memorised; they explain themselves.

Dates deliberately did not get an eighth hue

The date row uses a deep pine green close to the body text rather than another saturated hue — because **a date is an axis, not a category**. It is not the same kind of thing as the filters below it. Giving it a hue would imply it belongs to the same dimension as surface and source. Expressing a difference in information layer through a difference in colour *kind* is more effective than adding another divider line.

Two · Colour carries recognition; text only confirms

Each result card carries a vertical surface-coloured bar on its left edge. While scrolling quickly, **peripheral vision alone is enough to skip surfaces you do not want** — no need to read each label. The text label remains, but its role drops from recognition to confirmation.

This is a deliberate hierarchy decision: with twenty or thirty results on screen, moving recognition cost from *reading* to *perceiving* is the single largest improvement available to a list interface.

Three · Darkening the tennis yellow

The brand colour comes from the fluorescent yellow of a tennis ball. On the cream background (#FAF8F3) the raw fluorescent yellow floats, lacks contrast, and is unreadable at small text sizes. The shipped value is a darkened #C9DA2E , with a deeper #8FA318 reserved for situations needing more contrast.

When brand recognition and legibility conflict, legibility wins — but because the hue is preserved, nothing of the brand is actually lost.

Four · The paywall shows the product, not adjectives

The first thing on the subscription screen is not a feature list. It is **a card that looks exactly like a real notification**. Before deciding whether to pay, the user sees precisely what they are buying.

Because it looks real, the "Sample" badge cannot be omitted

The card carries an explicit "Sample" badge. Without it, someone would believe a court had actually opened and make a wasted trip — **realism buys trust, and trust must not be cashed in through misdirection**. The same badge, with the same wording, appears on the example card on the Alerts tab.

Five · A detail: the toggle knob is a tennis ball

Everyone has seen the default system switch. Replacing the knob with a tennis ball is the only purely decorative change in the entire app. It improves no metric — but it makes the product look like **something a person cared about**, rather than something assembled from a component library.

08

Design Highlights · Interaction

Good interaction design often shows up as something the user never noticed being handled.

One · No sign-up: the first screen is the product

The app opens directly on Find, ready to search. No registration, no login, no onboarding carousel. The user identifier is a random string generated locally on first launch and contains no personal information.

Rationale: the product proves its value within thirty seconds ("oh — there really is a court free at that hour"). Anything placed in front of those thirty seconds extracts a cost before the user has any reason to trust us.

Two · Search is free forever; only alerts are paid

This is not only pricing — it is trust design. **A user can verify, without paying anything, that our data is real, fresh and accurate.** Money is only discussed once they want the app to watch on their behalf.

The paywall states this plainly instead of hiding it: "Searching stays free. Alerts need a subscription."

Three · Data freshness is made explicit

The largest trust risk for any aggregator is "is what you are showing me stale?" The product handles this by saying so out loud:

- Every result set is labelled "**Updated X minutes ago**" — the latency is not hidden
- A "**Refresh now**" action lets the user force a real-time re-check at the decisive moment
- That action has a **cooldown**, so nobody assumes tapping repeatedly makes it more accurate
- Copy states that data comes from official public pages and **the official page has the final word**

Admitting a limitation builds trust

Writing "this may lag by a few minutes" into the interface is better than pretending the data is always live. If a user hits a stale result and the product never warned them, trust collapses in one step. If the product warned them in advance, the same event reads as normal.

Four · Reversibility: the address can be cleared

The home address is the only piece of personal information the user types. An early build could save it but not remove it — the save button was disabled on an empty field, so once stored, the address could not be deleted short of reinstalling.

Clearing the field now turns the button into "Clear". **Anything that can be entered must be revocable** — especially when it is personal and the user might hand their phone to someone else.

Five · Alert hours follow "when I can actually play"

Alert time ranges are not configured per alert. They are bound to a durable "when I can actually play" setting, defined per weekday and valid indefinitely.

Rationale: when a person can play is determined by work and life rhythm, and that is stable over a horizon of months. Making the user re-enter a stable fact each time an alert is created is the system offloading its own laziness onto them.

Six · Warnings appear only when they are relevant

Across 43 municipal centres and 202 individual courts in Paris, **exactly two are clay**. So when a user selects "clay", an explanation is essential — otherwise they will assume the app is broken.

But that warning **only appears when the municipal source and clay are both selected**. Showing it to someone who is not even looking at municipal courts is pure noise.

A warning under the right condition is help. The same warning under the wrong condition is interference. The more precisely the condition is written, the more the warning is worth.

Seven · The example card disappears once it has done its job

The Alerts tab opens with a "this is what a notification looks like" card explaining what the feature will actually deliver. Once the user **has received one real cancellation alert**, that card is removed permanently — and the flag never resets.

Rationale: teaching material is for people who do not yet understand. Someone who has received the real thing does not need it occupying a screen. Test notifications do not count, because they prove nothing about the live pipeline.

Eight · Fail open: better to let one through than to lock a payer out

Subscription status is queried locally against StoreKit. When the query fails — network hiccup, system error — the product **grants access** rather than blocking.

If failure blocks

One network hiccup shuts a paying customer out of what they bought, and they conclude their subscription is broken. That is an unacceptable failure mode.

If failure grants

In rare cases a non-subscriber briefly sees paid content. The cost is far lower, and in a production build the probability of query failure is very small.

Nine · Details are copied on hand-off

Tapping "Book on the official site" copies the court name and time slot to the clipboard and shows a short visual guide to what comes next on the other platform. This is **a design that admits its boundary** — we cannot control their interface, but we can minimise the cost of crossing into it.

09

Scope

P0 is "without this it does not exist". P1 is "this makes it better". P2 is "later".

P0 • Required for launch

#	Feature	Notes
F1	Home address	Geocoded for distance; editable and clearable
F2	Distance filter	1 / 3 / 5 / 10 / 20 / 40 km
F3	Date and hours filter	Next 7 days; draggable hour range
F4	Covered / open filter	Two independent toggles
F5	Surface filter	Clay / hard / synthetic grass
F6	Source filter	Municipal / Anybuddy / both
F7	Unified result list	Sort by distance, time or price; grouped by source
F8	Detail and hand-off	Detail + official link + auto-copy + guidance
F9	Cancellation alerts	Push on match; multiple monitors in parallel
F10	Saved courts	Follow the courts you use most
F11	Data freshness	"Updated X minutes ago" + manual refresh + cooldown
F12	Trilingual UI	French / English / Simplified Chinese, system-following with manual override
F13	Subscription and restore	Monthly / yearly + free trial + restore purchases

P1 • Second release

- Map view with court pins and distance rings
- Save filter combinations as one-tap presets
- Weather integration: rain probability flagged on open-air sessions
- Price comparison view across platforms for the same hour
- History: "this court was free three of the last four Wednesdays at 8pm"

P2 • Further out

- Android
- Apple Watch notifications and home-screen widgets
- Other cities, or other racquet sports (padel / squash)
- Availability prediction from accumulated history

Deliberately excluded

Automated booking (compliance and ethics), **partner matching** (unrelated to the core value, and Anybuddy's home turf), **user accounts** (the product works without one; with one comes data liability). Each exclusion is a decision, not an oversight.

10

Key Decisions and Trade-offs

The most useful part of a product document is usually not what was built, but why the alternatives were not.

One · Filter by straight-line distance, display cycling time

Decision

Filtering uses straight-line distance (free, instant). The maps API is called only for the handful of sessions a user actually opens.

Rationale

The Seine and the rail network make straight-line distance imprecise in Paris. But "1/3/5/10 km" is a coarse bucket and the error stays within it, while precise cycling time only matters for the two or three final candidates. Computing on demand cuts maps API usage by two orders of magnitude.

Two · Central crawling, not per-user crawling

Decision

The server crawls once; all users share the same snapshot.

Rationale

Cost and courtesy. Per-user crawling scales request volume linearly with user count, raising cost and putting unnecessary load on upstream sites. Central crawling decouples request volume from user growth.

Three · Entitlement is checked on device; the server never sees a receipt

Subscription status is queried directly against StoreKit — the answer is produced locally and signed by Apple. When there is no subscription, the product **revokes the push token from the server**, and the server can no longer send anything to that device.

This design removes an entire class of work

The server never needs to know who paid. Therefore **no receipt validation, no third-party subscription service, no payment data stored anywhere**. The access gate is not a branch in code — it is the physical fact of whether a token exists. Revoke the token and notifications stop by construction.

Four · Prices and trial length are read from StoreKit at runtime

Nothing on the paywall — price, billing period, free trial length — is hard-coded. Every value is fetched from StoreKit when the screen loads.

Rationale: Apple's pricing, exchange rates and tax treatment differ by country, so any hard-coded number eventually becomes a lie somewhere — and "displayed price does not match the charge" is an explicit rejection reason. By the same logic, if no free trial is configured in App Store Connect, the interface makes no mention of anything free: the button reads "Subscribe" and the auto-renewal disclosure switches to its no-trial variant.

Five · No account system

The user identifier is a random string generated on first launch. This means your configuration is lost when you switch phones — a real cost.

It is accepted because an account system brings registration friction, password recovery, deletion requests and GDPR obligations, while **the core value — receiving one notification at the right moment — does not require it at all.**

When the cost can be stated clearly and the benefit is plainly smaller than the burden, the right answer is not to build it.

11

Business Model

The free tier has to be genuinely useful. The paid tier has to be genuinely irreplaceable.

	Free	Subscription
Cross-platform search and filtering	Free forever	Included
Session detail and hand-off	Free forever	Included
Saved courts	Free forever	Included
Cancellation alerts	—	Core benefit
Multiple parallel monitors	—	Included

Pricing

€1.99

per month

€12.99

per year — 45% below monthly

7 days

free trial

The reference point is one municipal court booking (roughly €12–20). **The annual price is less than one and a half sessions**, an anchor that makes the decision almost calculation-free: one extra booking a year, won because of an alert, pays for the subscription.

Why search stays free forever

- **Verification comes first.** Users confirm the data is real before paying — the largest trust barrier for any aggregator;
- **Acquisition.** A free cross-platform search is itself worth sharing and recommending;
- **Clean value separation.** Free is "I could do this myself, just slowly." Paid is "I could not do this at all." Only the second sustains a subscription.

12

Data Flow and Architecture

The architecture serves two constraints: keep cost low, and do not burden upstream.

Pipeline

- 1 **Crawl** — the server reads public availability pages from both sources, roughly every 10 minutes
- 2 **Normalise** — two different schemas become one session model: court, coordinates, time, price, surface, cover, source
- 3 **Store** — current snapshot plus change history; the history is what makes cancellation detection possible
- 4 **Match** — newly available sessions are intersected with every user's monitor conditions
- 5 **Notify** — matches are delivered via push token, with a daily per-user cap

A cancellation is technically a diff

The core feature is not "query whether a court is free". It is "**detect a session that just changed from unavailable to available**". That is a set difference between two snapshots, not a query against one. This definition is why change history must be retained, and why crawl frequency is a product parameter rather than an implementation detail.

Client

Layer	Choice
Framework	React Native (Expo); build and submission via EAS
State and persistence	Local storage, no remote account
Subscriptions	StoreKit 2 via expo-iap; entitlement decided entirely on device
Notifications	Push token registered with our server, revoked when entitlement lapses
Localisation	Three dictionaries, system-following with manual override

Privacy minimisation

- **Not collected:** name, email, phone, third-party credentials
- **Transmitted:** home coordinates (for distance matching), push token, random identifier, filter preferences
- **Not retained:** the address string itself is used for real-time geocoding only and never stored
- **No tracking:** no advertising SDK, no analytics, no cross-app or cross-site tracking

13

Metrics

If only one number could be watched, it should be "how many bookings happened because of us".

North star

Bookings initiated from a notification hand-off

It is chosen because it simultaneously validates four things: the data was accurate, the match was right, the timing was useful, and the user was willing to act. If any link breaks, the number falls. Downloads, retention and subscription counts can all look healthy while the product creates no value at all.

Supporting metrics

Stage	Metric	What it watches
Discovery	Search → detail tap rate	Whether results are on target
Monitoring	% of users with an active monitor	Awareness and adoption of the core feature
Notification	Push open rate	Whether the notification is informative enough
Quality	Push → hand-off conversion	Whether we push sessions users do not want
Monetisation	Trial → paid conversion	Whether value lands within seven days
Retention	Renewal rate	Whether value is sustained rather than one-off

The number to guard is push → hand-off conversion

The dominant failure mode is not "too few notifications" — it is "**many notifications, none relevant**". Faced with an app that interrupts constantly and rarely helps, users do exactly one thing: revoke notification permission. Once revoked, the core value is zero and recovery is very unlikely. Hence a hard daily cap per user: better to miss one than to over-send.

14

Roadmap and Risks

Known risks, and the mitigations already built in.

Roadmap

Phase	Focus	Success criterion
Now • v1.0	Ship; validate that cancellation alerts create real value	A user books a court because of a notification
Next • v1.1	Map view, filter presets, weather integration	Search → detail tap rate improves
Then • v1.2	Availability prediction from history	Push → hand-off conversion improves
Later	Other cities / racquet sports / Android	The model replicates in a second city

Risks and mitigations

Risk	Impact	Mitigation
Upstream page structure changes and crawling breaks	High — data stops	Normalisation decoupled from crawling; freshness is user-visible, so failures surface rather than going silent
Upstream blocks crawling	High	Central crawling already minimises request volume; public data only, no login, no booking — leaving room for dialogue
Users revoke notification permission	High — core value goes to zero	Hard daily cap; push → hand-off conversion monitored as an early warning
Cancellations are rare in practice	Medium — thin core value	Monitoring 43 municipal centres — 202 courts — widens the sample; Anybuddy inventory supplements supply
Single-city ceiling	Medium	City is a configuration parameter rather than a hard-coded assumption

Closing

*What this product ultimately demonstrates is something small: in a domain where the information was already public, **reorganising that information is itself an act that creates real value**. The user still pays the official platform; the courts still belong to the city. The only thing that changed is that they finally know when one is free.*

Paris Tennis Dispo

巴黎公共网球场空位聚合与捡漏提醒

「有人取消了,现在空出来」—— 这条推送响起时,大概率全巴黎只有你知道。

文档版本
v1.0

产品版本
iOS 1.0.0

日期
2026 年 8 月

作者
刘嘉悦 Jiayue Liu

状态
已提交 App Store 审核

目录

01	摘要与产品定位	这个产品是什么,为谁做,以及为什么它值得存在。
02	问题:一次订场要试错多少次	三个叠加在一起的痛点,每一个单独看都不致命,合在一起足以让人放弃在巴黎打球。
03	核心洞察:两个战场	放弃抢票战场,不是退让,而是发现了一个几乎无人竞争的战场。
04	竞争格局与差异化	我们不与任何一方直接竞争 —— 最终成交仍然发生在他们的平台上。
05	目标用户与使用旅程	产品服务的不是「所有想打网球的人」,而是一类时间不固定、又不愿放弃打球的人。
06	解决方案总览	三个标签页,对应三种使用状态:主动找、被动等、管理自己。
07	设计亮点 · 界面	这一节的每一个决定都可以回答「为什么不是别的做法」。
08	设计亮点 · 交互	好的交互设计常常表现为「用户没有注意到有什么东西被处理掉了」。
09	功能范围	P0 是「不做就不成立」,P1 是「做了会更好」,P2 是「以后再说」。
10	关键决策与取舍	产品文档里最有价值的部分,往往不是做了什么,而是为什么没做别的。
11	商业模式	免费的部分要足够好用,付费的部分要足够不可替代。
12	数据流与技术架构	架构服务于两个约束:成本要低,以及不给上游添麻烦。
13	度量指标	如果只能看一个数字,那应该是「有多少次预订是因为我们才发生的」。
14	路线图与风险	已知的风险,以及现在就准备好的应对。

01

摘要与产品定位

这个产品是什么,为谁做,以及为什么它值得存在。

帮你在巴黎最快找到符合条件的空网球场,并在有人取消时第一时间告诉你 —— 然后一键跳到官方平台完成预订。

产品不代订、不存储用户的第三方账号密码、不挤占公共资源。它只做「发现」这一件事,并把它做到最好。

三句话说清

- 1. **问题:**巴黎的公共网球场预订网站只能一格一格点开才知道有没有空位,找一个合适的场次要试错几十次;而多数球友必须同时盯着市政和 Anybuddy 两个平台,来回切换。
- 2. **洞察:**真正的机会不在「放号瞬间」的抢票战场,而在「别人临时取消」腾出的空场 —— 它随机发生、全天任意时刻、几乎无人盯着。
- 3. **方案:**把两个平台的公开空场数据聚合成一个可筛选的列表,并在后台持续监控,一旦有符合你条件的场次出现就推送给你。

54

持续监控的场馆 (43 个市政中心 + 约 11 家 Anybuddy 俱乐部)

202

其中市政球场的片数

≈ 19,800

每轮扫描覆盖的「球场 × 日期 × 时段」可用性格子

10 分钟

数据自动刷新周期, 用户可手动触发实时复查

定位一句话

不是「订场工具」,而是「信息差工具」。用户最终仍然在官方平台完成预订 —— 我们卖的是「比别人早三十秒知道」。

为什么这个定位是刻意选的

项目最初的设想(v0.1)是代替用户自动下单抢场。在评估之后,我主动推翻了这个方向:

维度	v0.1 自动代订	v1.0 聚合 + 提醒
需要存用户密码	必须	完全不需要
GDPR 与数据安全	高风险,需法务介入	轻量:仅位置坐标与推送令牌
对方平台条款	明确违反反自动化条款	仅使用公开可见数据
App Store 上架	大概率被拒	成熟品类,可行
公共资源伦理	挤占他人机会	只是让信息更快到达
服务器成本	每个用户单独跑一份	中心化抓一次,全体共享

一次让项目从灰色地带走向可上架产品的转向。

这个取舍牺牲了「放号瞬间抢热门场」的能力,但换来了合规性、可持续性,以及一个更清晰的价值主张。第 03 节会说明为什么这个牺牲比看上去小得多。

02

问题:一次订场要试错多少次

三个叠加在一起的痛点,每一个单独看都不致命,合在一起足以让人放弃在巴黎打球。

痛点一 · 只有点开才知道有没有

巴黎市政的网球场预订网站采用「先选择、后揭示」的交互结构:你必须先指定球场、日期和具体时段,提交之后系统才告诉你这一格有没有空。**可用性信息藏在选择行为的后面,而不是摆在前面。**

于是找一个能打的场次变成这样一个循环:

- 1. 选一个球场
- 2. 选一个日期
- 3. 选一个时段
- 4. 提交,等待页面加载
- 5. 看到「无可用地」
- 6. 退回上一步,换一个时段

🔄 回到第 3 步 —— 每一次试错都是一次完整的页面往返,而你并不知道还要试多少次。

这个循环的代价可以被量化。以本产品监控的范围为例:

43	×	301	≈ 19,800
市政网球中心	天(可预订窗口)	次页面往返,才能把一周的 市政 场看完一遍	202 片球场 × 7 天 × 14 时段: 真正要被穷尽的可用性格子

问题的本质不是「慢」,是「不可穷尽」

用户真正需要的答案是「**未来七天里,离我 5 公里内、晚上、红土、有顶的场,什么时候有空**」。现有界面无法回答这个问题 —— 它只能回答「这一格有没有空」。用户被迫用几十次试错去逼近一个本该一次给出的答案,而且永远不确定自己有没有漏掉更好的选项。

痛点二 · 两个平台,来回切换

巴黎的网球场供给被切成两块,互不相通:

市政球场(tennis.paris.fr)

价格低(约 €12-20/小时),但热门时段极难订到,且只覆盖公立场地。

Anybuddy

覆盖私营俱乐部,空位相对多,但价格高出 3-4 倍。

现实中,一个每周想打一到两次的人**必须同时看这两个平台** —— 先在市政碰运气,没有再去 Anybuddy 补位。这意味着:

- 两套完全不同的筛选逻辑和界面语言,要在脑子里各维护一套心智模型

- 两次独立的试错循环,痛点一的成本直接翻倍
- **无法横向比较** —— 你没法在一个视图里看到「同一个晚上,市政 €20 和 Anybuddy €48 分别有哪些选择」
- 切换过程中容易丢失上下文:在 A 平台看好的时段,切到 B 平台又要重新设一遍条件

痛点三·网球在综合运动 App 里被稀释

Anybuddy 的产品定位是**综合球类运动的社交与预订平台** —— 它同时服务网球、羽毛球、壁球、匹克球、足球等多个项目,并且把「找球伴」作为核心场景之一。

这个定位对 Anybuddy 是合理的,但对一个只打网球的用户来说意味着:

- 每次进入都要先做一次「我要网球」的过滤动作,这个动作在高频使用下是纯粹的摩擦
- **网球特有的筛选维度被弱化** —— 场地材质(红土 / 硬地 / 人造草)和是否有顶,对网球用户是决定性的,对壁球用户毫无意义,因此不会被放在信息层级的前排
- 社交功能占据了界面的重要位置,而只想约固定球伴打球的人并不需要它

我们的切口

从一个横向平台里,把纵向的那一层抽出来重做。 这一点在实现上是字面意义的 —— 抓取程序读的是每个俱乐部的 /tennis 子页面,匹克球、羽毛球、壁球的库存从一开始就不进我们的数据。我们只取两个来源里的网球场数据,按网球用户真正在乎的维度(距离、时段、材质、有顶、价格、来源)重新组织信息层级 —— 让那些在综合平台里被压到第三层的筛选条件,变成第一屏就能拨动的开关。

三个痛点的叠加效应

单看每一个痛点,用户都能忍。但它们是相乘而非相加的关系:一次试错循环的成本 × 两个平台 × 每次都要重新过滤运动类型,结果是在巴黎找一个满意的网球场,所需的注意力成本高到让很多人干脆放弃临时打球的念头,只保留固定的包场时间。

产品要消灭的不是某一次点击,而是「算了,今天不打了」这个念头。

03

核心洞察:两个战场

放弃抢票战场,不是退让,而是发现了一个几乎无人竞争的战场。

「订到热门场」看上去是一件事,拆开之后其实是**两个完全不同的战场**,竞争强度相差一个数量级。

战场 A · 放号瞬间

市政系统在每天固定时刻释放新的可预订时段。几百人同时刷新,热门场次在几秒内被抢空。

- 竞争者数量:数百人同时在线
- 决胜因素:网络延迟与手速,或者自动化脚本
- 结论:在不使用自动下单的前提下,这个战场赢不了。主动放弃。

战场 B · 别人取消腾出的空场

有人临时有事退订,那个时段会重新回到系统里变成可用。这件事的特征完全不同:

特征	战场 A 放号	战场 B 捡漏
发生时刻	每天固定,人尽皆知	全天随机,无法预期
同时竞争者	数百	接近于零
决胜因素	手速与延迟	谁先知道
人力能否覆盖	能(定好闹钟)	不能 —— 没人会一天刷两百次网页
服务器能否覆盖	能	能,而且轻松

这就是产品的杀手锏

人不可能一天刷两百次网站,但服务器可以每十分钟扫一遍全城。「明天晚上 8 点,Élisabeth 红土室内有人取消了,现在空出来」—— 这条推送在你手机上响起的时候,大概率全巴黎只有你知道。这个价值一点不比自动抢低,而且完全合规。

为什么这个洞察改变了产品形态

一旦确认战场 B 才是机会所在,产品的重心就从「搜索」转向了「监控」:

如果重心是搜索

产品是一个更好用的目录。用户主动打开、查询、离开。
价值随使用频率线性增长,而使用频率天然很低。

如果重心是监控

产品在用户不打开它的时候仍然在工作。价值发生在推送响起的那一刻 —— 那是一个用户自己不可能创造的时刻。

这直接决定了三件事:捡漏提醒被做成核心页而不是附属功能;搜索永久免费,只有提醒收费(见第 11 节);以及推送的质量控制成为产品最重要的工程问题(见第 08 节)。

04

竞争格局与差异化

我们不与任何一方直接竞争 —— 最终成交仍然发生在他们的平台上。

	tennis.paris.fr	Anybuddy	Paris Tennis Dispo
覆盖范围	仅市政公立场	仅签约私营俱乐部	两者聚合
运动项目	网球等市政项目	多项球类运动	只做网球
可用性一览	需逐格点开	列表式,较好	跨平台统一列表
按材质 / 有顶筛选	有限	层级较深	第一屏开关
取消捡漏提醒	无	无	核心功能
价格横向比较	不可能	仅自身库存	同屏对比
最终预订	自有	自有	跳转官方完成

我们与上游是互补而非替代

产品不接管交易、不抽佣、不存储对方的账号凭据。用户在我们这里完成「发现」,在官方平台完成「预订」。这既是合规上的必要选择,也让产品避开了与上游平台的直接对抗。

为什么这个位置是可防守的

- 上游没有动机做这件事 —— 市政系统没有做体验优化的商业压力;Anybuddy 没有动机把用户导向竞品的库存。
- 聚合本身构成壁垒 —— 价值来自「同时看两边」,只做单边的产品无法复制这个价值。
- 捡漏提醒有时间积累效应 —— 持续的历史数据可以支撑未来的预测能力(见第 14 节路线图)。
- 垂直深度 —— 只做网球,意味着可以把材质、有顶、球场朝向这类细节做到综合平台不可能做的程度。

05

目标用户与使用旅程

产品服务的不是「所有想打网球的人」,而是一类时间不固定、又不愿放弃打球的人。

核心用户画像

维度	描述
是谁	在巴黎生活的业余网球爱好者,有一到两个固定球伴
频率	每周想打 1-2 次
约束	工作时间不固定,无法提前很久锁定时段;预算敏感但非不惜代价
现有行为	在市政网站反复试错,失败后转向 Anybuddy 补位;经常因为「找起来太麻烦」而放弃
未被满足的需求	「告诉我什么时候有空场,而不是让我一个个去问」

旅程 A · 随手找场

- 1 打开 App,直接进入「找场」页 —— 无需注册、无需登录
- 2 设置出发地址(仅首次),用于按距离筛选
- 3 拨动筛选:距离 / 日期 / 时段 / 材质 / 有顶 / 来源
- 4 在同一个列表里看到市政与 Anybuddy 的结果,按距离、时间或价格排序
- 5 点开某个场次查看详情:球场名、地址、骑行时间、价格、材质
- 6 点「去官方预订」→ 跳转对应平台,关键信息已自动复制到剪贴板

旅程 B · 捡漏提醒(核心)

- 1 在「捡漏」页创建监控条件(可同时开多个)
- 2 设定「我能打球的时间」 —— 按星期几分别设置时段
- 3 后台持续扫描,一旦出现匹配的空场立即推送
- 4 推送到达 → 点击直接进入该场次详情
- 5 跳转官方平台完成预订

两条旅程的关系

旅程 A 是**获客与信任建立** —— 它永久免费,让用户在付费之前就获得真实价值。旅程 B 是**价值兑现与变现** —— 它提供的是用户自己无论如何都做不到的事。前者证明数据是真的,后者才值得付费。

06

解决方案总览

三个标签页,对应三种使用状态:主动找、被动等、管理自己。

标签页	解决的问题	核心交互
找场	痛点一 + 二 + 三:把近两万个可用性格子压缩成一张可筛选的列表	六个维度的筛选开关 + 统一结果列表 + 三种排序
捡漏	战场 B:在用户不看手机的时候持续工作	监控条件配置 + 可打球时间段 + 推送
设置	出发地、语言、通知、订阅状态	低频但必须可达

信息架构的一个关键决定

「找场」和「捡漏」共用同一套筛选维度(距离、时段、材质、有顶、来源),但它们是**两组独立的状态**。

为什么不共用一组条件

「我现在想找什么」和「我希望被提醒什么」是两种不同的意图。前者是当下的、临时的;后者是长期的、稳定的。共用会导致一次随手搜索污染长期的提醒规则。

代价与补偿

代价是用户要配置两次。补偿是「捡漏」的条件配置被刻意做得更简单 —— 时段用「我能打球的时间」按星期几设置一次,长期有效,不需要每次调整。

设计亮点 · 界面

这一节的每一个决定都可以回答「为什么不是别的做法」。

一、七个筛选维度,七种颜色,同屏不打架

「找场」页需要同时呈现六到七组分类标签:场地材质三种、有顶 / 露天两种、数据来源两种,再加上日期。如果配色随手挑,同屏出现时会互相干扰,用户必须逐条读文字才能分辨。

因此配色不是逐个挑的,而是按色相环等距排布的:



- 前五个色相在环上大致等距,任意两个相邻标签的色相差都足够大,不需要看文字就能区分;
- 来源的两个颜色刻意落在紫与玫红这两个「剩下的空档」上,避免与材质色撞车;
- 语义自带记忆点:红土用陶土色、人造草用草绿、露天用日晒的金色、有顶用背阴的靛蓝 —— 颜色不需要被记忆,它自己会说话。

日期刻意没有拿到第八个色相

日期条用的是接近正文的深松绿,而不是一个新的鲜艳色相。因为日期是「轴」,不是「类别」—— 它和下面那几排筛选不是同一种东西。给它一个色相,会让用户误以为它和材质、来源是并列的维度。用颜色的「不同类」来表达信息结构的「不同层」,比加一条分隔线更有效。

二、色彩承担识别负担,文字只做确认

结果列表里每张卡片左侧有一条材质色的竖条。用户在快速滚动时靠余光就能筛掉不想要的材质,不需要逐条阅读标签文字。文字标签仍然保留,但它的角色从「识别」降级为「确认」。

这是一个刻意的信息层级设计:在一屏可能有二三十条结果的场景下,把识别成本从「阅读」降到「感知」,是列表体验最大的一次提升。

三、网球黄的压深处理

品牌色取自网球的荧光黄,但原始的荧光黄放在米白底(#FAF8F3)上会发飘、对比度不足、且在小尺寸文字上不可读。因此实际使用的是压深一档的 #C9DA2E ,并为需要更强对比的场景准备了更深的 #8FA318 。

品牌识别度与可读性冲突时,可读性优先 —— 但通过保留色相不变,品牌感并没有损失。

四、付费墙上摆的是产品本身,不是形容词

订阅页的第一屏不是功能罗列,而是一张长得和真推送一模一样的通知样卡。用户在决定是否付费之前,先看见自己买的到底是什么。

正因为它长得像真的,「示例」角标就不能省

样卡右上角有一个明确的「示例」标记。没有它,会有用户以为真有场空出来了,然后白跑一趟 —— 这是用真实感换来的信任,不能用误导来兑现。同样的角标也用在捡漏页的示例卡上,两处说法一致。

五、细节:开关滑块是一颗网球

系统默认的开关谁都见过。把滑块换成一颗网球,是全 App 唯一一处纯装饰性的改动 —— 它不提升任何效率,但它让这个 App 看起来是有人在乎地做出来的,而不是组件库拼出来的。

08

设计亮点 · 交互

好的交互设计常常表现为「用户没有注意到有什么东西被处理掉了」。

一、零注册:第一屏就是产品本身

打开 App 直接进入「找场」页,可以立刻搜索。没有注册、没有登录、没有引导页。用户标识是首次启动时本地生成的随机串,不含任何个人信息。

理由:产品的价值在三十秒内就能被验证(「哦,原来这个时间真的有场」)。任何挡在这三十秒前面的东西,都是在用户还没相信产品之前就索取成本。

二、搜索永久免费,只有提醒收费

这不只是定价策略,更是一个信任设计:用户可以在完全不付费的情况下,验证我们的数据是真的、是新的、是准的。等到他真的想让 App 替他盯着的时候,再谈钱。

付费墙上写着这句话,而不是把它藏起来——「搜索球场始终免费。提醒需要订阅。」

三、数据新鲜度被显式化

聚合类产品最大的信任风险是「你给我的数据是不是过期的」。产品的处理是把这件事摊开来讲:

- 每次结果都标注「**X 分钟前更新**」,不回避延迟的存在
- 提供「**立即刷新**」按钮,让用户在关键时刻可以主动触发一次实时复查
- 刷新按钮带**冷却时间**,防止用户误以为「多点几次会更准」而反复触发
- 文案明确说明数据来自官方公开页面,官方页面拥有最终解释权

承认局限反而建立信任

把「可能有几分钟延迟」写在界面上,比假装数据永远实时要好。用户一旦遇到一次不同步而产品此前从未提示过,信任会一次性崩塌;而如果提前说明,同样的情况只会被理解为「正常范围」。

四、可逆性:出发地可以被清空

出发地址是产品里唯一一处用户输入的个人信息。早期版本只能保存、不能删除——输入框为空时保存按钮是禁用的,导致地址一旦存进去,除了重装 App 没有任何办法拿掉。

现在清空输入框后,按钮会变成「清除」。**任何一个能被输入的东西,都必须能被撤销**——尤其当它是个人信息、而用户可能要把手机递给别人看的时候。

五、时段筛选跟着「我能打球的时间」走

捡漏提醒的时段不是每次单独设置,而是绑定到一个长期的「我能打球的时间」配置——按星期几分别设定,一次设好长期有效。

理由:一个人能打球的时间是由工作和生活节奏决定的,它在几个月的尺度上是稳定的。让用户每次创建提醒时重复输入这个稳定的事实,是把系统的懒惰转嫁给了用户。

六、提示只在相关的时候出现

全巴黎 43 个市政中心、202 片球场里,只有 2 片是红土。因此当用户勾选「红土」时,必须给一个说明,否则他会以为 App 坏了。

但这条提示只在「同时选中了市政来源 + 红土」时才出现。用户如果压根没在看市政场,弹这条提示只是噪音。

条件正确的提示是帮助;条件不对的提示是干扰。判断条件写得越细,提示的价值越高。

七、示例卡在完成使命之后永久消失

捡漏页顶部有一张「推送长这样」的示例卡,用来向新用户说明这个功能到底会给他什么。一旦用户真的收到过一条捡漏推送,这张卡片就永久消失 —— 而且这个状态一旦置位就不再回退。

理由:教学内容是讲给还不懂的人听的。已经收到过真货的人不需要它再占着一屏。测试推送不计入,因为它没有证明真实链路可用。

八、失败开放:宁可漏放,不可错锁

订阅状态由手机本地向 StoreKit 查询。当查询失败时(网络抖动、系统异常),产品选择放行而不是拦截。

如果失败时拦截

一次网络抖动就会把付过钱的用户挡在门外,并且他会以为自己的订阅出了问题。这是不可接受的体验损伤。

如果失败时放行

极少数情况下会有未订阅用户短暂看到付费内容。代价远小于前者,且在正式包中查询失败的概率极低。

九、跳转官方时自动复制关键信息

点击「去官方预订」时,球场名与时段被自动复制到剪贴板,并配有一张指引图说明在对方平台上接下来点哪里。这是一个承认边界的设计 —— 我们无法控制对方的界面,但可以把跨越边界的那一步成本降到最低。

09

功能范围

P0 是「不做就不成立」,P1 是「做了会更好」,P2 是「以后再说」。

P0 · 首个版本必须有

#	功能	说明
F1	出发地设置	地址转经纬度,用于距离计算;可修改、可清空
F2	距离筛选	1 / 3 / 5 / 10 / 20 / 40 km
F3	日期与时段筛选	未来 7 天;时段区间可拖动调整
F4	有顶 / 露天筛选	两个独立开关,可同时选
F5	场地材质筛选	红土 / 硬地 / 人造草
F6	来源筛选	市政 / Anybuddy / 全部
F7	统一结果列表	按距离、时间或价格排序;市政与 Anybuddy 分组展示
F8	场次详情与跳转	详情 + 跳转官方 + 自动复制 + 操作指引
F9	捡漏提醒	后台匹配即推送;可同时开多个条件
F10	收藏球场	常去的球场单独关注
F11	数据新鲜度	「X 分钟前更新」+ 手动刷新 + 冷却
F12	三语界面	法语 / 英语 / 简体中文,跟随系统并可手动切换
F13	订阅与恢复购买	月付 / 年付 + 免费试用 + 恢复购买

P1 · 第二版

- 地图视图:球场落点与距离圈
- 常用筛选组合保存为一键预设
- 天气联动:露天场次提前提示降雨概率
- 价格对比视图:同一时段跨平台的价格排序
- 历史统计:「某球场周三 20:00 过去四周有三次空出来」

P2 · 更远

- Android 版本
- Apple Watch 通知与桌面小组件
- 扩展到其他城市,或其他球拍类运动(padel / squash)
- 基于历史数据的空场概率预测

被刻意排除在外的

代订与自动下单(合规与伦理)、**球伴匹配**(与核心价值无关,且是 Anybuddy 的主场)、**账号体系**(没有它产品也成立,有了它就要承担数据责任)。每一项排除都是主动的,不是没想到。

关键决策与取舍

产品文档里最有价值的部分,往往不是做了什么,而是为什么没做别的。

决策一·用直线距离筛选,用骑行时间展示

决定

筛选阶段用直线距离(免费、即时);只对用户真正点开的那几个场次调用地图 API 计算骑行时间。

理由

巴黎有塞纳河与铁路切割,直线距离确有误差。但「1/3/5/10 km」本身就是粗档位,误差在档位内可以接受;而精确的骑行时间只对最终候选的两三个场次有意义——按需计算可以把地图 API 的用量降低两个数量级。

决策二·中心化抓取,而非每用户抓取

决定

服务器统一抓取一次,所有用户共享同一份数据。

理由

成本与礼貌。若每个用户独立抓取,请求量随用户数线性增长,既推高成本,也对上游站点构成不必要的压力。中心化抓取让请求量与用户规模解耦。

决策三·订阅校验放在设备本地,服务器不碰收据

订阅状态由手机直接向 StoreKit 查询——答案由苹果在本地给出并签名。没有订阅时,产品把推送令牌从服务器撤销,服务器从此发不出任何东西给这台设备。

这个设计消掉了一整类工作

服务器完全不需要知道谁付了钱,因此**不需要验证收据、不需要接第三方订阅管理服务、不需要存储任何支付相关数据**。权限闸门不是一段判断逻辑,而是「令牌在不在」这个物理事实——令牌一撤,推送自然停。

决策四·价格与试用信息全部运行时取自 StoreKit

付费墙上的价格、订阅周期、免费试用长度,没有一项写死在代码里,全部在运行时向 StoreKit 查询。

理由:苹果在各国的定价、汇率、含税与否都不同,写死的数字一定會在某个国家变成谎话;而「显示的价格与实际扣款不符」是明确的拒审理由。同理,如果后台没有配置免费试用,界面上就不会出现任何提及免费的字样——按钮直接变成「订阅」,自动续订说明也换成不含试用的版本。

决策五·不做账号体系

用户标识是首次启动时本地生成的随机串。这意味着换手机会丢失配置——一个真实的代价。

接受这个代价的理由是:账号体系会带来注册摩擦、密码找回、数据删除请求、GDPR 责任等一整套负担,而**产品的核心价值(在正确的时刻收到一条推送)完全不需要它**。当代价可以被清晰陈述、且收益明确小于负担时,就应该不做。

11

商业模式

免费的部分要足够好用,付费的部分要足够不可替代。

	免费	订阅
跨平台搜索与筛选	永久免费	包含
场次详情与跳转预订	永久免费	包含
收藏球场	永久免费	包含
捡漏推送提醒	—	核心权益
多条监控条件	—	包含

定价

€1,99

每月

€12,99

每年 —— 相比月付省 45%

7 天

免费试用

参照系是一次市政场地的费用(约 €12-20)。**年费低于一次半的场地费**,这个锚点让付费决策变得几乎不需要计算 —— 只要一年里因为提醒多订到一次场,订阅就回本了。

为什么把搜索永久免费

- **验证成本前置** —— 用户在付费前就能确认数据真实可用,这是聚合类产品最大的信任门槛;
- **获客成本** —— 免费的搜索本身就是一个值得被分享和推荐的工具;
- **价值分离清晰** —— 免费的是「我自己也能做,只是慢」;付费的是「我自己无论如何做不到」。后者才撑得起订阅。

数据流与技术架构

架构服务于两个约束:成本要低,以及不给上游添麻烦。

数据流

- 1

抓取

—— 服务端定时读取两个来源的公开可用性页面(约每 10 分钟一轮)
- 2

归一化

—— 把两套结构不同的数据统一成同一个场次模型:球场、坐标、时间、价格、材质、有顶、来源
- 3

存储

—— 保留当前快照与变更历史;变更历史是捡漏识别的基础
- 4

匹配

—— 新出现的空场与所有用户的监控条件求交集
- 5

推送

—— 命中即通过推送令牌下发;每人每天有上限,避免打扰
- 「捡漏」在技术上是一次差分

产品的核心功能不是「查询有没有空场」,而是「发现一个刚刚从没有变成有的场次」。它是两次快照之间的差集,而不是某一次快照的查询结果。这个定义决定了必须保留变更历史,也决定了抓取频率是产品的关键参数而非实现细节。

客户端

层面	选择
框架	React Native (Expo),构建与提交通过 EAS
状态与持久化	本地存储,无远程账号
订阅	StoreKit 2,通过 expo-iap 封装;权限判定完全在本地
推送	推送令牌注册到自有服务端,订阅失效时主动撤销
国际化	三语词典 + 跟随系统语言,可手动覆盖

隐私最小化

- **不收集**:姓名、邮箱、电话、第三方账号密码
- **会传输**:出发地坐标(用于距离匹配)、推送令牌、随机用户标识、筛选偏好
- **不留存**:地址原文仅用于实时换算坐标,不入库
- **不追踪**:无广告 SDK、无分析统计、不跨 App 或跨站追踪

度量指标

如果只能看一个数字,那应该是「有多少次预订是因为我们才发生的」。

北极星指标

通过推送产生的跳转预订次数

选择它的理由:它同时包含了「数据是准的」「匹配是对的」「时机是及时的」「用户是愿意行动的」四重验证。任何一环失效,这个数字都会掉下来。而单看下载量、留存或订阅数,都可能在产品实际没有创造价值的情况下依然好看。

支撑指标

环节	指标	关注什么
发现	搜索到详情的点击率	筛选结果是否切题
监控	创建了监控条件的用户占比	核心功能的认知与采纳
推送	推送打开率	推送内容是否足够有信息量
推送质量	推送到跳转的转化率	是否在推送「用户其实不想要的场次」
变现	试用转付费率	价值是否在 7 天内被兑现
留存	订阅续费率	价值是否持续,而非一次性

最需要盯的是「推送到跳转的转化率」

这个产品最大的失败模式不是「推送太少」,而是「**推送太多但不相关**」。用户对一个持续打扰又帮不上忙的 App 只会做一件事:关掉通知权限 —— 而通知权限一旦被关,产品的核心价值就归零了,且极难挽回。因此每人每天的推送有硬上限,宁可漏掉,不可滥发。

路线图与风险

已知的风险,以及现在就准备好的应对。

路线图

阶段	重点	成功标准
现在 · v1.0	上架,验证捡漏推送的真实价值	有用户因为推送真的订到了场
下一步 · v1.1	地图视图、筛选预设、天气联动	搜索到详情的点击率提升
之后 · v1.2	基于历史数据的空场概率预测	推送到跳转的转化率提升
更远	其他城市 / 其他球拍运动 / Android	模式在第二个城市被验证可复制

风险与应对

风险	影响	应对
上游页面结构变更导致抓取失效	高 —— 数据断流	归一化层与抓取层解耦;数据新鲜度对用户可见,异常会先被用户察觉而非静默失败
上游封禁抓取行为	高	中心化抓取已将请求量降到最低;仅使用公开数据,不登录、不代订,保留沟通与合作的空间
推送权限被用户关闭	高 —— 核心价值归零	每日推送上限;监控「推送到跳转的转化率」作为早期预警
市政系统的取消行为本身很稀少	中 —— 核心价值稀薄	监控范围覆盖 43 个市政中心共 202 片球场以扩大样本;同时聚合 Anybuddy 库存作为补充供给
单一城市天花板	中	架构上把「城市」做成配置而非硬编码,为复制到第二个城市留出空间

结语

这个产品最终要证明的是一件很小的事:在一个信息本来就是公开的领域里,把信息重新组织一遍,本身就能创造真实的价值。用户仍然在官方平台付款,球场仍然属于城市 —— 唯一改变的是,他终于知道了什么时候有空。