Suivi et analyse des interactions sur une plateforme

Projet BD NoSQL

Manal Ez-zaher
Anthony Menghi
Mehdi Ghoulam

21 Novembre 2024

Contents

1 Introduction

Dans ce projet, nous avons pour objectif de concevoir une base de données MongoDB permettant de suivre et d'analyser les interactions des utilisateurs de Mindlet. L'objectif principal est d'exploiter ces données pour extraire des insights pertinents et éclairer les décisions stratégiques, notamment en matière d'amélioration de l'expérience utilisateur et d'évaluation de la performance des fonctionnalités.

La plateforme intègre plusieurs points de contact, tels qu'une application mobile, un site web et un système de support client. Ces différents canaux génèrent des interactions variées, allant de l'utilisation des fonctionnalités premium à la soumission de tickets de support, en passant par la modification des cartes générées par l'IA.

2 Identification des interactions pertinentes

Nous devions dans un premier temps identifier des interactions pertinentes en lien avec notre application.

Voici une liste de ces interactions :

Interactions sur le site web

- Temps passé sur le site : Permet de mesurer l'engagement des visiteurs.
- Inscriptions à la newsletter : Mesure de la conversion des visiteurs en abonnés réguliers.
- Nombre de clics sur le site : Permet de mesuer les actions faites par le visiteur.

Interactions avec le support client

- Ouverture de tickets : Identifie les problèmes ou demandes les plus fréquents.

Interactions liées aux abonnements

- Nombre d'abonnements par type (standard, premium) : Analyse des préférences des utilisateurs pour les différentes offres.
- Renouvellements : Indique si l'utilisateur renouvelle son abonnement.
- Nombre de désabonnements : Permet d'analyser les raisons et le moment où les utilisateurs quittent le service.

Interactions sur l'application mobile

- Temps passé sur l'application : Mesure l'engagement et l'utilisation globale.
- Nombre de clics sur les différentes fonctionnalités : Permet d'identifier les fonctionnalités les plus utilisées.

- Modifications des cartes générées par IA : Évalue la pertinence de l'IA et son impact sur l'expérience utilisateur.
- Utilisation des fonctionnalités premium : Indique le taux d'adoption des options avancées.

Les interactions identifiées offrent une base solide pour analyser le comportement des utilisateurs et améliorer les services proposés. Voici leur utilité détaillée :

1. Fiabilité de l'IA dans la génération des cartes
   En suivant l'interaction "modifications des cartes générées par IA", nous pouvons évaluer la pertinence des suggestions automatiques. Un taux élevé de modifications pourrait indiquer des problèmes dans l'efficacité de l'IA, nécessitant des ajustements.
2. Conversion et engagement via le site web
   Le suivi des "temps passés sur le site" et des "inscriptions à la newsletter" permet de mesurer l'attractivité de notre plateforme et d'identifier des leviers pour augmenter la conversion des visiteurs en utilisateurs.
3. Efficacité du support client
   Les "tickets support" triés par catégories (bugs, demandes de fonctionnalités, etc.) et par utilisateur permettent d'identifier les zones de friction ou les besoins d'amélioration prioritaires. Par exemple, un nombre élevé de tickets liés à une même fonctionnalité peut signaler une expérience utilisateur défaillante.
4. Adoption et fidélisation via les abonnements
   Les "types d'abonnements" et les données sur les "renouvellements" et "désabonnements" sont cruciales pour comprendre la rétention des utilisateurs et les raisons de leur départ. Par exemple, un désabonnement fréquent après un certain temps d'utilisation pourrait pointer des failles dans les offres ou les fonctionnalités.
5. Performance de l'application mobile
   Les interactions comme "temps passé sur l'application" ou "utilisation des fonctionnalités premium" permettent de mesurer l'engagement des utilisateurs et d'identifier les fonctionnalités qui captivent le plus ou le moins.

3 Conception de la base de données MongoDB

3.1 Choix de la Modélisation

Nous avons opté pour une modélisation basée sur plusieurs collections dans MongoDB comme le montre le schéma figure 1. Ce choix repose sur les raisons suivantes :

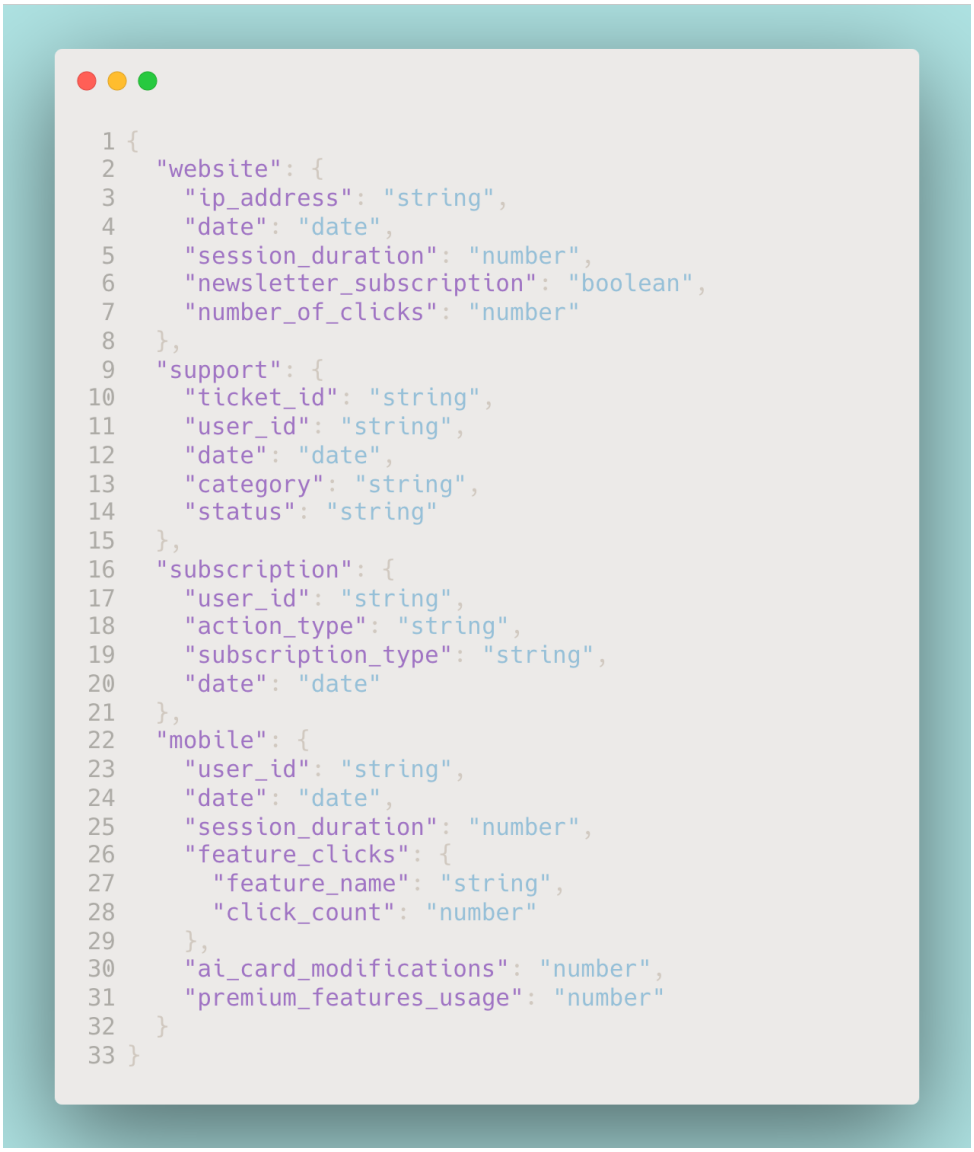
1. Structure des données
   Les interactions proviennent de différentes sources et ont des structures de données variées. Les interactions liées au support incluent des catégories de tickets, tandis que celles de l'application se concentrent sur les clics ou le temps d'utilisation. Utiliser une seule collection entraînerait une complexité inutile en raison de champs très variés.

2. Éviter la duplication
   Une collection unique nécessiterait de stocker plusieurs champs vides pour chaque type d'interaction, entraînant une duplication massive. La séparation en collections permet de mieux optimiser l'utilisation du stockage.

3. Facilité les requêtes et agrégations
   Avec plusieurs collections, les données peuvent être interrogées de manière indépendante selon le domaine. Cela simplifie l'écriture des requêtes analytiques et améliore leur performance, en limitant les traitements aux données pertinentes.

4. Lisibilité et maintenance
   Une modélisation en collections distinctes améliore la lisibilité de la base de données et facilite sa maintenance. Chaque collection est spécialisée pour un domaine, ce qui réduit les risques d'erreurs lors des manipulations ou analyses.

3.2 Modélisation



```
{
  "website": {
    "ip_address": "string",
    "date": "date",
    "session_duration": "number",
    "newsletter_subscription": "boolean",
    "number_of_clicks": "number"
  },
  "support": {
    "ticket_id": "string",
    "user_id": "string",
    "date": "date",
    "category": "string",
    "status": "string"
  },
  "subscription": {
    "user_id": "string",
    "action_type": "string",
    "subscription_type": "string",
    "date": "date"
  },
  "mobile": {
    "user_id": "string",
    "date": "date",
    "session_duration": "number",
    "feature_clicks": {
      "feature_name": "string",
      "click_count": "number"
    },
    "ai_card_modifications": "number",
    "premium_features_usage": "number"
  }
}
```

Figure 1: Schéma des collections MongoDB

3.3 Details des champs

Website Cette collection regroupe les informations relatives aux visiteurs du site web. Elle inclut des données telles que l'adresse IP, la date de visite, la durée de la session, ainsi que des informations sur l'interaction avec le formulaire de newsletter et les sections visitées, avec un suivi du nombre de clics.

Support Cette collection contient des données sur les tickets soumis par les utilisateurs. Elle enregistre la catégorie du ticket (signalement de bug, demande de nouvelles fonctionnalités, etc.), son statut (nouveau, en cours, traité), l'utilisateur à l'origine de la demande, ainsi que la date de création du ticket.

Subscription Cette collection se concentre sur les abonnements à l'application. Elle détaille les types d'actions effectuées, qu'il s'agisse d'un nouvel abonnement, d'un renouvellement ou d'une annulation, ainsi que la catégorie d'abonnement et la date associée.

Mobile Enfin, cette collection capture les interactions liées à l'application mobile. Elle inclut des données telles que les zones de navigation, avec le nombre de clics, la durée des sessions, la fréquence de modification des cartes générées par l'IA.

4 Requêtes avec agrégations

4.1 Nombre totale d'abonnements par type

```
use("tp_mongodb");

db.subscription.aggregate([
  {
    $group: {
      _id: "$subscription_type",
      total: { $sum: 1 }
    }
  },
  { $sort: { total: -1 } }
]);
```

Figure 2: Nombre totale d'abonnements par type

4.2 Total des tickets par catégorie et par jour

```
use("tp_mongodb");

db.support.aggregate([
  {
    $group: {
      _id: {
        category: "$category",
        date: {
          $dateToString: {
            format: "%Y-%m-%d",
            date: "$date"
          }
        }
      },
      totalTickets: { $sum: 1 }
    }
  },
  { $sort: { "_id.week": 1 } }
]);
```

Figure 3: Total des tickets par catégorie et par jour

4.3 Temps moyen passé sur le site par semaine

```
use("tp_mongodb");

db.website.aggregate([
  {
    $group: {
      _id: {
        $dateToString: {
          format: "%Y-%m-%d",
          date: "$date"
        }
      },
      avgTimeSpent: { $avg: "$session_duration" }
    }
  },
  { $sort: { "_id": 1 } }
]);

```

Figure 4: Temps moyen passé sur le site par semaine

4.4 Fonctionnalités les plus cliquées

```
use("tp_mongodb");

db.mobile.aggregate([
  { $unwind: "$feature_clicks" },
  {
    $group: {
      _id: "$feature_clicks.feature_name",
      totalClicks: { $sum: "$feature_clicks.click_count" }
    }
  },
  { $sort: { totalClicks: -1 } }
]);
```

Figure 5: Fonctionnalités les plus cliquées

4.5 Proportion des utilisateurs qui modifient des cartes générées par l'IA

```
use("tp_mongodb");

db.mobile.aggregate([
  {
    $group: {
      _id: "$user_id",
      totalModifications: { $sum: "$ai_card_modifications" }
    }
  },
  {
    $group: {
      _id: null,
      totalUsers: { $sum: 1 },
      usersWithModifications: {
        $sum: {
          $cond: [{ $gt: ["$totalModifications", 0] }, 1, 0]
        }
      }
    }
  },
  {
    $project: {
      _id: 0,
      totalUsers: 1,
      usersWithModifications: 1,
      proportionModified: {
        $multiply: [{ $divide: ["$usersWithModifications", "$totalUsers"] }, 100]
      }
    }
  }
]);
```

Figure 6: Proportion des utilisateurs qui modifient des cartes générées par l'IA

5 Comparaison avec une approche relationnelle

Il existe une différence importante entre les SGBD relationnels et non relationnels, notamment en ce qui concerne les types de données à stocker. Dans notre cas, nous collectons des interactions, ce qui diffère des données relationnelles classiques où l'objectif principal est souvent les opérations de mise à jour régulières. Ici, il s'agit plutôt de collecter des données éphémères, utilisées pour des calculs ou des analyses.

En d'autres termes, chaque SGBD a ses avantages et ses inconvénients, et le choix dépend surtout du type de données à enregistrer. Avec les SGBD non relationnels, comme MongoDB, on bénéficie d'une grande liberté dans la définition des schémas, ce qui est idéal pour le stockage d'interactions. Les champs peuvent changer, rester vides ou évoluer facilement, sans avoir à reconfigurer toute la base.

En revanche, les SGBD relationnels, comme PostgreSQL, sont plus adaptés aux données structurées et bien définies. Ils sont parfaits pour gérer des informations telles que les utilisateurs, les abonnements ou les relations entre différentes entités, car ces données nécessitent des relations clairement établies et une cohérence stricte.

Ainsi, le choix entre les deux types de SGBD dépend avant tout des besoins spécifiques du projet et du type de données à manipuler.